# STARK

## Stellar Tools from raw sequencing data Analysis to variant RanKing



# GUIDE UTILISATEUR

version 0.9.17b

# INTRODUCTION

STARK est un environnement d'analyse de données de séquençage conçu pour des données de santé (recommandations ANPGM/INCa) suivant les recommandations internationales et nationales, dont l'objectif principal est l'aide à l'interprétation des résultats en vue d'un diagnostic clinique. Flexible et adapté aux besoins des biologistes, efficient en terme de consommation de ressources, sont approche « application » permet une stabilité, un suivi et une maîtrise des analyses.

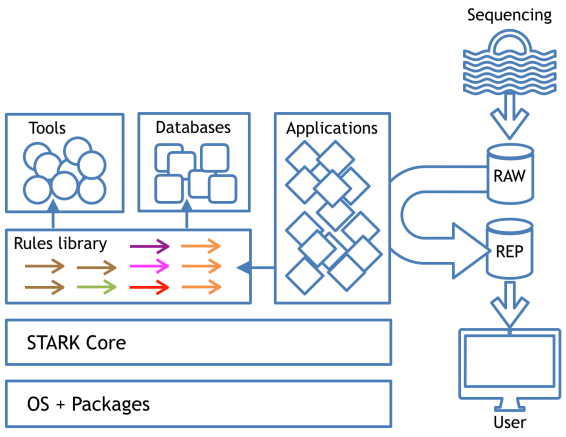
STARK permet l'analyse des données de séquençage brutes (BCL, FASTQ, BAM, SAM CRAM) et génère des fichiers résultats annotés aux formats VCF et TSV (inter-opérabilité avec des interfaces de visualisation/interprétation, tableur) et des rapports (format PDF et texte, résumé des résultats, contrôle qualité, traçabilité des analyses).

STARK peut analyser les données de séquençage de 2 façons différentes :

- L'analyse d'un séquençage Illumina (« RUN ») complet utilise les fichiers BCL générés par les séquenceurs et les fichiers de configuration (SampleSeet, fichiers de marqueurs), ainsi que le fichier contenant les régions d'intérêts (Manifest, optionnel).
- L'analyse d'échantillons indépendants à partir de fichiers contenant les reads séquencés (format standards FASTQ, BAM, SAM ou CRAM pour l'archivage), et d'un fichier contenant les régions d'intérêts (format BED ou Manifest Illumina, optionnel).

STARK effectue les étapes de demultiplexage, d'alignement, de post alignement (marquage des duplicats, réalignement et recalibration des BAM, clipping des primers), détection de variants (calling), d'annotation des variants et de priorisation des variants (en fonction de leur qualité, pathogénicité probable, intérêt dans la pathologie...). L'ensemble de ces étapes est effectué par une constellation d'outils de référence (e.g. bcl2fastq, FastQC, GATK, Picard, Samtools, Bcftools, VarScan) et complémentaires les uns des autres, pour des analyses spécifiques (application), à chaque pathologie, technologie, type de données, pratique.

Le code de STARK est développé sur le principe du Makefile, qui est de construire un graphe de dépendance des fichiers à générer, en définissant des règles permettant de générer ces fichiers. Ces règles constituent des briques indépendantes pouvant être utilisées pour définir des applications. Cette philosophie permet une utilisation simple du code une évolution facilité par l'ajout de règles supplémentaires. Également, ces règles peuvent être exécutées en parallèle sur plusieurs processeurs, permettant une exécution efficiente de l'analyse en terme de ressource (scalable et distribué). Le développement de STARK tend à suivre les bonnes pratiques de développement définies par la communauté nationale (INCa).




Les applications disponibles ont été développés et testés pour l'analyse des données générées par la technologie capture ou amplicon, avec les kits Illumina, Multiplicom et Agilent. Ces applications peuvent détecter les mutations (substitutions ou indels) constitutionnelles ou somatiques, les mosaïques à faible fréquence allélique, les CNV. Ces applications sont basées sur les recommandations de la communauté scientifique nationale (INCa et ANPGM) et les recommandations internationales (GATK).

# INSTALLATION

Afin de faciliter son déploiement, STARK est disponible en VM, en Docker, et en TARball pour une installation compète de STARK « from scratch » . Dans chacun des cas, la distribution contient la liste d'outils (binaires) et de bases de données nécessaires à sont fonctionnement.

- Certains outils sont sous licence (e.g. GATK, ANNOVAR).

## PRÉREQUIS

**CPU** : Au minimum 2 CPUs sont requis et idéalement 8 CPUs sont recommandés.

**RAM** : Au minimum 5Go de RAM par CPU et idéalement 32Go de RAM. L'application est particulièrement gourmande en mémoire lors de l'alignement des séquences par BWA et l'annotation des variants par SnpEff, en raison de la mise en mémoire du génome et des bases de données d'annotation.

**ESPACE DISQUE** : Au minimum 500Go d'espace est nécessaire pour stocker les outils et les bases de données, pour créer un espace temporaire d'analyse, et accueillir un minimum de données brutes et de données résultats.

- Dimension du nombre de CPU, de la RAM et de l'ESPACE DISQUE en rapport avec l'activité (nombre d'échantillons, panel de gènes, exomes, genomes, …).
- Des montages externes sont possibles, notamment pour les espaces des données brutes et des données résultats qui augmenterons au fur et à mesure de l'activité.

## INSTALLATION

Les distributions de STARK sont disponibles en s'adressant à Contact. Pour plus de détails et d'autres options d'installation, se référer à Installation avancée.

### *Installation via VM*

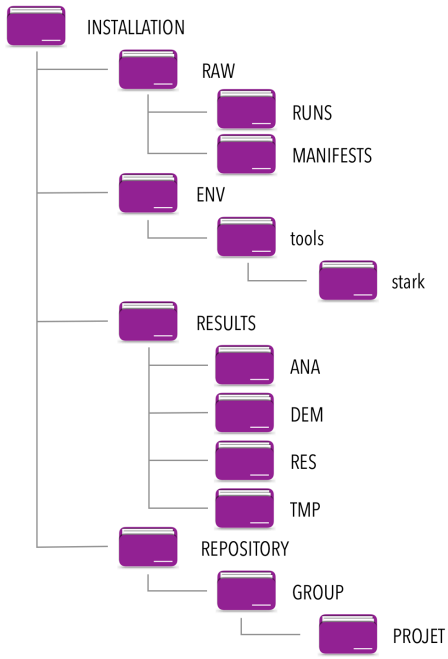
STARK est disponible via une machine virtuelle, intégrant l'ensemble du système (OS CentOS 6.7, modules système, outils d'analyse, bases de données, génomes), pré-configuré et re-configurable. Cette méthode permet de proposer un environnement stable et maîtrisé. Cette machine virtuelle peut être déployée sur un système virtualisé, sur un serveur physique, ou par l'utilisation d'outils de virtualisation (VirtualBox, VMWare).

### *Installation via Docker*

STARK est disponible en Docker. Ce choix nécessite l'installation de docker sur votre machine. Le docker a été testé avec un socle CentOS 7.

Le Docker contient STARK et les outils binaires. Les génomes et bases de données (p.e. ANNOVAR et SNPEff) sont disponibles en « TARball » et sont mappés lors du lancement du docker. Un script est disponible afin de faciliter l'installation et l'utilisation de STARK en docker (cf. Contact).

## CONFIGURATION




Les données bruts de séquençage sont déposées dans RAW. Les analyses sont traitées dans le dossier RESULTS en respectant la mise en forme suivante : ANA - log des analyses; DEM - demultiplexing; RES - résultats d'analyse; TMP - fichiers temporaires. Un espace REPOSITORY met à disposition les résultats d'analyse.

La configuration générale permet une installation des différents espaces disques.

Le fichier de configuration par défaut :

```
<INSTALLATION>/<ENV>/tools/stark/current/bin/env.sh
```

Ce fichier contient les paramètres principaux de STARK. Il définit les chemins des outils (ENV), le dossier de traitement des résultats (RESULTS), le dossier contenant les données brutes des runs (RAW) et le dossier temporaire où travailler. Il détermine ensuite les sous-dossiers où trouver les génomes, les résultats, les logs, les manifests, etc …

▸ Exemple d'installation

Modifier le fichier env.sh pour intégrer les choix d'installation :

```
# FOLDERS
############
FOLDER_ENV=<INSTALLATION>/ENV                #Emplacement du dossier ENV
FOLDER_RUN=<INSTALLATION>/RAW/RUNS           #Emplacement des données brutes
FOLDER_MANIFEST=<INSTALLATION>/RAW/MANIFESTS #Emplacement des manifests
FOLDER_RESULTS=<INSTALLATION>/RESULTS        #Emplacement où sont généré les résultats
FOLDER_REPOSITORY=<INSTALLATION>/REPOSITORY  #Possibilité d'un repository (optionnel)
```

À ce stade la configuration de STARK est par défaut. Pour une utilisation plus approfondie, STARK permet d'utiliser des applications pré-configurées ou de créer et d'utiliser ses propres applications. Cette étape permet de redéfinir les paramètres par défaut (cf Application).

# GET STARTED

Il est possible de lancer une analyse STARK sur un ou plusieurs échantillons (FASTQ) ou bien directement sur un run. Cette dernière permettant une analyse complète des runs en sortie de séquenceur (BCL).

*Les fichiers minimum requis pour lancer une analyse*

| Fichier | Description |
|---|---|
| fastq ou bam | STARK permet de démarrer une analyse à partir de fichiers fastq, bam, sam, cram.<br>*Obligatoire* |
| bed ou manifest | L'analyse STARK repose sur l'utilisation d'un manifest. Il peut également générer un manifest à partir du bed.<br>*Défaut: analyse sur génome complet.* |
| env.<APPLICATION>sh | Ce fichier comporte l'ensemble des paramètres à utiliser pour démarrer une analyse avec l'application APPLICATION.<br>*Défaut: env.sh* |
| SampleSheet.csv | Fichier à déposer dans la racine du répertoire du run.<br>*Obligatoire uniquement dans le cas d'une analyse par run* |

Les différentes options disponibles sont détaillées avec la commande suivante :

```
<INSTALLATION>/ENV/tools/stark/current/bin/STARK.sh —help
```

| Fichier | Description | |
|---|---|---|
| `-f/--fastq` | `<STRING1,STRING2,...>` | La liste des FASTQ/BAM/SAM/CRAM<br>Formats : *fastq.gz, *fastq.gz, *bam, *ubam, *cram, *ucram, *sam, *usam |
| `-r/--runs` | `<STRING1,STRING2,...>` | La liste des runs à analyser |
| `-e/--env/--app` | `<FILE>` | Le fichier envirronement à utiliser pour la configuration de l'application |
| `-l/--samplesheet` | `<FILE>` | format SampleSheet.csv d'illumina |
| `-i/--application_infos` | | Informations sur les applications disponibles |
| `-y/--pipelines_infos` | | Informations sur les pipelines et leurs composants (e.g. aligners, callers…) |
| `-g/--release_infos` | | Informations sur la version de STARK et des différentes étapes des pipelines |
| `-x/—tools_infos` | | Informations sur les outils configuré dans STARK |

## LANCER UNE ANALYSE PAR SAMPLE

Exemple de commande

```
<INSTALLATION>/<ENV>/tools/stark/current/bin/STARK.sh -f SAMPLE_A.R1.fastq.gz,SAMPLE_B.R1.fastq.gz -q
SAMPLE_A.R2.fastq.gz, SAMPLE_B.R2.fastq.gz --bed region.bed --output <OUTPUTDIR> --env
env.<APPLICATION>.sh
```

La configuration utilisée par défaut est celle du fichier env.sh. il est possible de configurer sa propre application ou d'utiliser une application pré-configurée (cf. Application).

## LANCER UNE ANALYSE PAR RUN

L'analyse par run est conçue pour permettre de traiter les données brut directement en sortie de séquenceur. Dans ce cas il est nécessaire de préciser la SampleSheet ou de la déposer dans le dossier du run.

Exemple de commande :

```
<INSTALLATION>/<ENV>/tools/stark/current/bin/STARK.sh -r RUN
```

L'application peut être renseignée en paramètre de la commande :

```
<INSTALLATION>/<ENV>/tools/stark/current/bin/STARK.sh -r RUN1 -e env.<APPLICATION>.sh
```

Ou dans le fichier SampleSheet.csv du run (en en-tête par défaut, ou pour chaque échantillon)

```
[Header]
Investigator Name,APPLICATION
…
[Data]
Sample_ID,…,Sample_Project
SAMPLE1,…,APPLICATION,…
SAMPLE2,…,APPLICATION,…
SAMPLE3,…,APPLICATION,…
…
```

## UTILISATION D'UNE APPLICATION PRÉ-CONFIGURÉE

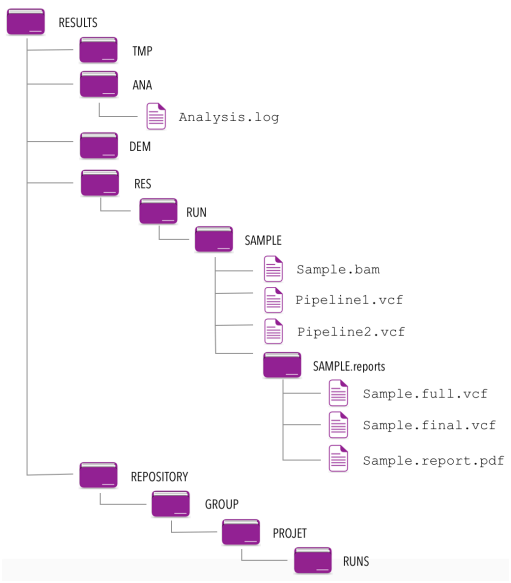
Il est également possible d'utiliser des applications pré-définies, intégrant des pipelines pré-configurés pour certaines applications (e.g. Hematologie, Tumeur Solide, BRCA Constitutionnel, BRCA somatic, Oncogénétique, Maladies Rares, Exome, Exome somatic, Genome, CNV, Mosaic, Amplicon, Capture).

*Liste de certaines applications disponibles :*

| Fichier | Description |
|---|---|
| env.sh | Environnement par défaut avec les paramètres de base |
| env.CANOE.sh | L'application CANOES utilise l'outil CANOES sur un ensemble de données générées par la technologie Capture pour identifier les CNV. Cette application nécessite plusieurs échantillons (techniquement au moins 3, statistiquement au moins 30, cf publication), et a été validée sur plusieurs sets de données (identification de CNV constitutionnels) |
| env.ITD.sh | L'application ITD utilise le caller ITDSeek sur des données générées par la technologie Amplicon. Elle identifie les mutations dans le gène FLT3 : « Improves the detection of FLT3-internal tandem duplication (ITD) of varying length,position and at low allelic burden » |
| env.HEMATOLOGY.sh | Identification de mutations somatiques (peu connues), grande clonalité possible et faible quantité de cellules cancéreuses (VAF>= 2%) sur panel de gènes et données d'Amplicon (Laboratoire d'hématologie, kit Illumina - TruSight Myeloid Sequencing Panel) |
| env.ONCOGENET.sh | Identification des mutations constitutionnelles (SNV et CNV) sur petit panel de gènes et données de capture (Laboratoire d'oncogénétique, Cancer du sein, kit Agilent et kit SophiaGenetics) |
| env.EXOME.sh | Identification des mutations constitutionnelles sur panel full génome sur données de capture |
| env.EXOME_SOMATIC.sh | Identification des mutations constitutionnelles (SNV et CNV) sur panel full exome sur données de capture |
| env.GENOME.sh | Identification des mutations somatiques sur panel full exome (haute profondeur) sur données de capture |
| env.GERMLINE.sh | Identification des mutations constitutionnelles (SNV et CNV) sur grand panel de gènes et données de capture (Laboratoire de génétique, Maladies rares, kit Agilent) |
| env.SOLIDTUMOR.sh | Identification de mutations somatiques (VAF>= 4%) sur panel de gènes et données d'Amplicon (Laboratoire de biologie moléculaire, kit Illumina - TruSight Myeloid Sequencing Panel) |

# RÉSULTATS

## ARCHITECTURE




Les résultats d'analyse sont accessibles dans le dossier RESULTS définit dans le fichier env.sh ou dans le repertoire définit avec l'option --output. Les résultats sont déposés selon la structure suivante.

### *Structure des dossiers*

Les résultats du demultiplexing (fastq)

```
<INSTALLATION>/<RESULTS>/DEM
```

Les résultats de l'analyse (BAM, VCF, metrics)

```
<INSTALLATION>/<RESULTS>/RES
```

Les fichiers log

```
<INSTALLATION>/<RESULTS>/ANA
```

## ACCÈS AUX FICHIERS

Les résultats d'analyses sont disponibles pour chaque patient dans le dossier suivant :

```
<INSTALLATION>/<RESULTS>/<RUN>/<SAMPLE>/
```

Les rapports d'analyses sont disponibles dans un sous-dossier :

```
<INSTALLATION>/<RESULTS>/<RUN>/<SAMPLE>/<SAMPLE>.reports/
```

Les dossiers FASTQC et METRICS permettent d'accéder aux contrôles qualité des données (qualité de séquençage, couverture, stats). Les dossiers *.fastqc donnent accès au contrôle qualité des reads séquencés (fastq) et les dossiers *.metrics aux contrôles qualité des BAMs et VCFs.

```
<INSTALLATION>/RESULTS/<RUN>/<SAMPLE>/*.fastqc/
<INSTALLATION>/RESULTS/<RUN>/<SAMPLE>/*.metrics/
```

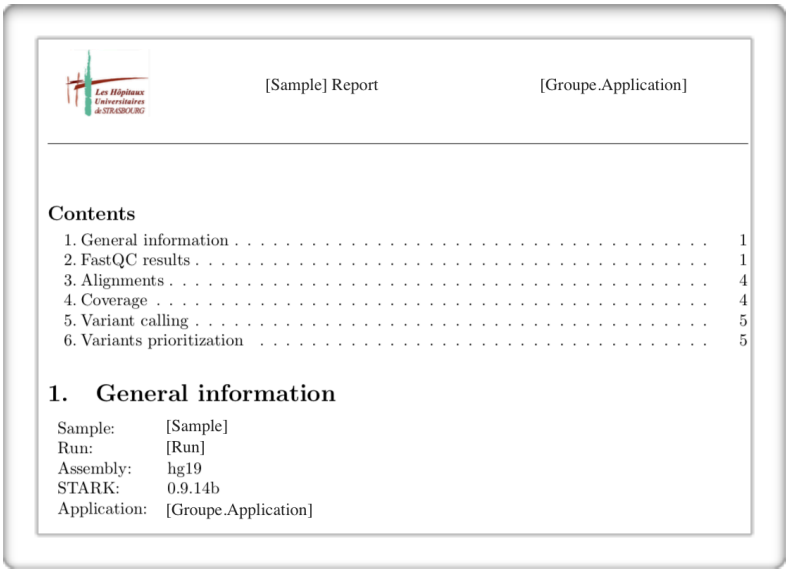
Il est également possible de déposer les résultats dans un REPOSITORY. Les résultats sont alors organisés en fonction du groupe et du projet associé à l'application. Les fichiers « final.vcf », « full.vcf » et « stark.report.pdf » sont directement accessibles dans la racine du dossier <SAMPLE>. L'ensemble des résultat est disponible dans le dossier DATA. La copie des fichiers est assurée par le programme rsync (par boucle continue) et un contrôle d'intégrité des fichiers de type checksum MD5.

```
<INSTALLATION>/<REPOSITORY>/<GROUP>/<PROJECT>/<RUN>/<SAMPLE>/<SAMPLE>.final.vcf
<INSTALLATION>/<REPOSITORY>/<GROUP>/<PROJECT>/<RUN>/<SAMPLE>/<SAMPLE>.full.vcf
<INSTALLATION>/<REPOSITORY>/<GROUP>/<PROJECT>/<RUN>/<SAMPLE>/<SAMPLE>.stark.report.pdf
<INSTALLATION>/<REPOSITORY>/<GROUP>/<PROJECT>/<RUN>/<SAMPLE>/DATA/*
```

*Les rapports en sortie d'analyse*

| Fichier | Description |
|---|---|
| *.full.vcf | Le fichier VCF contenant les résultats de tous les pipelines utilisés dans l'analyse (une colonne par pipeline). |
| *.final.vcf | Le fichier VCF contenant le 'merge' de tous les pipelines (une colonne unique pour tous les pipelines). |
| *.stark.report.pdf | Le rapport de l'analyse |

## RAPPORT D'ANALYSE



[Sample] Report [Groupe.Application]

Contents



1. General information

| | |
|---|---|
| Sample: | [Sample] |
| Run: | [Run] |
| Assembly: | hg19 |
| STARK: | 0.9.14b |
| Application: | [Groupe.Application] |

- Un rapport est généré automatiquement en fin d 'analyse (ci-contre, exemple de rapport). Il donne un aperçu des données générées à chaque étape de l'analyse et une liste de variants filtrés et priorisés. Les paramètres définis par l'APPLICATION utilisée (défaut: env.sh) sont également précisés, notamment le fichier de configuration des régions d'intérêt (manifest et/ou bed).

Comment suivre le déroulement de l'analyse ?

Fichier l'analyse :

`<INSTALLATION>/<RESULTS>/ANA/*.log`

Suivre le démultiplexing :

`<INSTALLATION>/<RESULTS>/DEM/*.log`

Suivre la génération des fichiers VCFs et du rapport :

`<INSTALLATION>/<RESULTS>/RES/<RUN>/*.log`

- Faire un `tail -f <log>` du fichier log pour suivre l'analyse

Les fichiers .log permettent de savoir si un run s'est bien déroulé. Ils sont disponibles à plusieurs étapes de l'analyse. Une erreur fatale au pipeline est décrite par le préfixe *** .

- Faire un `grep "\*\*\*"` du fichier log pour identifier les erreurs

# OPTIONS D'ANALYSE

Plusieurs options sont disponibles pour démarrer une analyse par sample ou par run. L'analyse par sample est pensée pour une analyse rapide et flexible de données de séquençage (FASTQ, BAM). L'analyse par run permet une analyse complète et facilitée des données brutes en sortie de séquenceur.

```
<INSTALLATION>/<ENV>/tools/stark/current/bin/STARK.sh --help
```

## OPTIONS D'ANALYSE PAR UNE LISTE D'ÉCHANTILLONS

*Les options disponibles*

| Option | Description | Description |
|---|---|---|
| -f/--fastq/--fastq_R1 | <STRING1,STRING2,...> | La liste des FASTQ/BAM/SAM/CRAM<br>Formats : *fastq.gz, *fastq.gz, *bam, *ubam, *cram, *ucram, *sam, *usam |
| -q | <STRING1,STRING2,...> | La liste des fastqR2 (si –f = liste de fastqR1) |
| -e/--env/--app | <FILE> | Le fichier à utiliser pour la configuration de l'application |
| -b/--bed | <STRING1,STRING2,...> | la liste des fichier BED correspondant |
| -j/--genes | <STRING1,STRING2,...> | La liste des "fichiers gènes" en format BED |
| -m/—transcripts | <STRING1,STRING2,...> | La liste des "fichiers transcripts" en format TSV |
| -s/--sample | <STRING1,STRING2,...> | La liste des samples correspondants |
| -r/--run | <STRING1,STRING2,...> | La liste des runs correspondants |
| -o/--output/--results | <FOLDER> | Le dossier résultats |
| -u/--repository | <FOLDER> | Le dossier repository |
| -p/--pipelines | <STRING1,STRING2,...> | La liste des pipelines |
| -t/--threads | <INTEGER> | Le nombre de CPU à utiliser |
| -g/--by_sample | | parallélisation de l'analyse par échantillon |
| -a/--adapters | <STRING> | Les adaptateurs à rétirer |
| -v/--verbose | | option verbose |
| -d/--debug | | option debug |
| -n/--release | | option release |

## OPTIONS D'ANALYSE PAR RUNS

⚠ Ce choix d'analyse requiert que le fichier SampleSheet du run soit défini en paramètre ou déposée dans le dossier du run.

*Les options disponibles*

| Option | Format | Description |
|---|---|---|
| `-r/--runs` | `<STRING1,STRING2,...>` | La liste des runs à analyser |
| `-e/--env/--app` | `<FILE>` | Le fichier à utiliser pour la configuration de l'application |
| `-a/--aligners` | `<STRING1,STRING2,...>` | La liste des aligners |
| `-c/--callers` | `<STRING1,STRING2,...>` | La liste des callers |
| `-k/--annotators` | `<STRING1,STRING2,...>` | La liste des annotateurs |
| `-p/--pipelines` | `<STRING1,STRING2,...>` | La liste des pipelines (priorité par rapport aux listes d'aligners, callers et annotateurs) |
| `-f/--filter_samples` | `<STRING1,STRING2,...>` | échantillons à utiliser pour l'analyse |
| `-l/--samplesheet` | `<FILE>` | format SampleSheet.csv d'illumina |
| `-z/--parallelization` | | parallélisation de l'analyse par run |
| `-b/--by_sample` | | parallélisation de l'analyse par échantillon |
| `-v/--verbose` | | option verbose |
| `-d/--debug` | | option debug |
| `-n/--release` | | option release |

# APPLICATION

Les configurations de STARK se font via des fichiers d'environnement (e.g. env.sh). STARK permet une configuration spécifique à une application (exemple : panel de gènes, exome, …) via la création de fichiers environnements dédiés. Cette configuration permet de définir les paramètres par défaut à utiliser lors du lancement d'une analyse pour une application. Cette étape est nécessaire pour permettre l'automatisation des analyses en sortie de séquenceur.

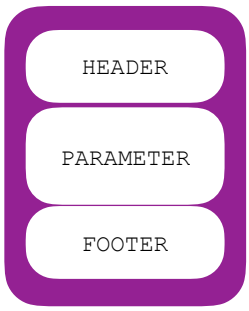
## STRUCTURE D'UNE APPLICATION

Le nom du fichier:

```
<INSTALLATION>/<ENV>/tools/stark/current/bin/env.<APPLICATION>.sh
```

Il est possible de redéfinir la structure des dossiers propres à un groupe et un projet (e.g. les dossiers des résultats, des données brutes, des fichiers temporaires), ainsi que les pipelines à utiliser (aligners, callers et annotators). Ce fichier définit aussi les étapes supplémentaires à effectuer (e.g. post-alignement, marquage des duplicates, calcul des metrics), les annotations, et différentes options.

La structure d'une application est la suivante :




Paramètres à importer :
```
source env_header.sh
```

Configurations des paramètres propre à une application
```
#configuration de l'application
```

Vérification des paramètres. <Ne pas modifier>
```
source env_footer.sh
```

## CRÉER SA PROPRE APPLICATION

Pour permettre une grande souplesse d'utilisation, Il est également possible de créer sa propre application pour modifier les paramètres par défaut, c'est à dire redéfinir l'ensemble des paramètres utilisés dans STARK (outils, pipelines) :

```
<INSTALLATION>/<ENV>/tools/stark/current/bin/env.<APPLICATION>.sh
```

### *A. Définir l'application*

Si souhaité, il est possible de récupérer les paramètres par défaut (cf Héritage des applications). Par exemple :

```
source $STARK_FOLDER/env.sh
source $STARK_FOLDER/env.MY_GROUP.sh
```

Définir le nom de l'application permet de détecter l'application dans la SampleSheet à travers l' « investigator name ».

```
APP_NAME="MY_APP"
```

Des règles spécifiques à l'application, chargées automatiquement, peuvent être créées dans le dossier :

```
<INSTALLATION>/<ENV>/tools/stark/current/bin/<MY_APP>.rules.mk/
```

Définir groupe et projet permet de structurer le dépôt des données dans le REPOSITORY :

```
GROUP="MY_APP"
PROJECT="2017"
```

### B. Définir les dossiers

Emplacement des données brutes (raw data)

```
FOLDER_RUN=<INSTALLATION>/RAW/RUNS
```

Emplacement des fichiers manifest et manifest.genes

```
FOLDER_MANIFEST=<INSTALLATION>/RAW/MANIFESTS
```

Emplacement des résultats. Toutes les données générées vont respecter la mise en forme suivante :

- RES : les résultats d'analyses (BAM, VCF, metrics)
- DEM : les résultats du démultiplexage
- ANA : les fichiers log des analyses
- TMP : les fichiers temporaires

```
FOLDER_RESULTS=<INSTALLATION>/RESULTS
```

Emplacement des outils: tous les outils utilisés par STARK et autres (dossier ENV)

```
FOLDER_ENV=<INSTALLATION>/ENV
```

Emplacement du REPOSITORY. Les résultats peuvent être déposés dans un dossiers (un montage externe par exemple). Laisser vide si vous ne désirez pas faire de copie.

```
FOLDER_REPOSITORY=<INSTALLATION>/REPOSITORY
```

### C. Définir les pipelines

Définir le génome de référence :

```
ASSEMBLY=hg19
```

Les pipelines à utiliser pour l'analyse :

Si les variables ALIGNERS, CALLERS et ANNOTATORS sont définies (voir ci-dessous), la variable PIPELINES est générée automatiquement. Cette variable peut également servir à ajouter un pipeline additionnel à ceux générés par la combinaison ALIGNERS, CALLERS et ANNOTATORS (cf Liste des aligneurs-callers-annotateurs). Si tout de même aucune variable n'est définie, le pipeline par défaut est utilisé (si défini dans env.sh).

```
PIPELINES="ALIGNER1.CALLER1.ANNOTATOR ALIGNER1.CALLER2.ANNOTATOR1 ALIGNER2.CALLER1.ANNOTATOR1"
```

Aligneurs utilisés pour l'analyse. (e.g. bwamem bwasw bwaaln)

```
ALIGNERS="bwamem"
```

Callers utilisés pour l'analyse. (e.g. gatkHC gatkUG VarScan samtools)

```
CALLERS="gatkHC_DIAG_IP gatkUG_DIAG_IP"
```

Annotateurs utilisés pour l'analyse. (e.g. howard snpeff)

```
ANNOTATORS="howard"
```

Il est possible de prioriser l'affichage des pipelines dans le rapport (final.vcf)

```
PRIORITIZE_PIPELINES_LIST="bwamem.gatkHC_DIAG_IP.howard,bwamem.gatkUG_DIAG_IP.howard"
```

### D. Définir les paramètres

Paramètre pour le demultiplexage

```
BARCODE_MISMATCHES=1
```

Génération des Bam Metrics (1/TRUE/YES/Y ou 0/FALSE/NO/N). Consommation de temps et de mémoire. Switcher entres les deux pour les exome/genome pour de meilleures performances.

```
BAM_METRICS=0
```

Recalibration des variants après le Calling (1/TRUE/YES/Y or 0/FALSE/NO/N). Si la recalibration échoue (principalement due à un manque de données pour générer des statistiques), aucune action ne sera faite.

```
VARIANT_RECALIBRATION=0
```

L'interval padding (par défaut 0), permet d'inclure dans l'analyse les régions autours des zones définies dans le manifest (nous recommandons environs 100 paires de bases).

```
INTERVAL_PADDING=50
```

Le critère de couverture est utilisé pour calculer le pourcentage de bases au dessus du critère de couverture. (e.g. 1,30,100 pour calculer sur 1X, 30X et 100X).

```
COVERAGE_CRITERIA="1,30,100"
```

Le nombre de bases à analyser autour des exons définis dans le fichier BED.

```
NB_BASES_AROUND=20
```

Pour les calcules des metrics, il est nécessaire de définir le fichier en format bed contenant 5'UTR, 3'UTR et les régions génomiques codantes.

```
BEDFILE_GENES="project.bed"
```

Pour vérifier les BAMs à toutes les étapes de la manipulation (clipping, réalignement, …). Gourmand en temps de calcul, mais stoppe l'analyse en cas de reads manquants.

```
BAM_CHECK_STEPS=1
```

Les étapes de post alignement (par défaut: « sorting realignment clipping compress »). Comprend toutes les étapes entre l'alignement et le calling des variants.

Format: « step1 step2 step3 »

Exemple: « sorting realignment clipping compress » (<ALIGNER>.compress.clipping.realigned.sorting.bam)

Le fichier BAM va être : 1-sorted, 2-realigned, 3-clipped et 4-compressed

Les étapes sont définies par des règles makefile, et sont disponibles par la commande:

```
$STARK/STARK.sh --pipelines_infos
```

les étapes disponibles:

- sorting: sorting du BAM
- compress: compression du BAM (voir la variable $BAM_COMPRESSION)
- realignment: réalignement local
- markduplicates: BAM Mark Duplicates
- clipping: Clipping des BAM par rapport aux primers définies dans le fichier manifest

Les étapes généralement utilisées:

- « sorting realignment clipping compress » pour la technologie Amplicon
- « sorting realignment markduplicates compress » pour la technologie Capture

```
POST_ALIGNMENT_STEPS="sorting markduplicates realignment compress"
```

Niveau de compression final du BAM (unaligned.bam, ALIGNER.bam)

```
BAM_COMPRESSION=5
```

Le nombre de threads utilisés pour l'analyse (AUTO va considérer « CORE-1 » threads). Le nombre de threads doit être compris entre 1 et le total des CPUs disponibles sur la machine (auto-ajustement si la valeur ne correspond pas).

```
THREADS=AUTO
```

### E. Configuration HOWARD

HOWARD est un outil d'annotation de VCF intégré dans STARK, permettant l'annotation des variants, des calculs sur ces annotations, la priorisation des variants suivant des filtres prédéfinis, ainsi qu'une conversion au format TSV des VCF. HOWARD est disponible en tant qu'annotateur dans les pipelines, et est utilisé pour l'annotation des fichiers VCF associés au rapport. Pour plus d'information sur le paramétrage d'HOWARD, veuillez vous référer à la documentation spécifique d'HOWARD disponible dans le dossier ENV.

Les paramètres suivants permettent de choisir les annotations des annotateurs « howard » et « howard_minimal » et des fichiers « final.vcf » et « full.vcf ».

L'annotation de l'annotateur « howard » :

```
HOWARD_ANNOTATION="core,snpeff_split"
```

L'annotation de l'annotateur « howard_minimal » :

```
HOWARD_ANNOTATION_MINIMAL="core,snpeff_split"
```

L'annotation des VCF du rapport final :

```
HOWARD_ANNOTATION_REPORT="core,frequency,score,annotation,prediction,snpeff,snpeff_hgvs,snpeff_split"
```

De nombreux calculs peuvent être effectués lors de l'utilisation d'HOWARD. Les paramètres suivants permettent de choisir les calculs des annotateurs « howard » et « howard_minimal », ainsi que lors de l'annotation des VCF associés au rapport.

Calculs de l'annotateur « howard » :

```
HOWARD_CALCULATION="VARTYPE,NOMEN"
```

Calculs de l'annotateur « howard_minimal » :

```
HOWARD_CALCULATION_MINIMAL="VARTYPE,NOMEN"
```

Calculs des VCF du rapport final :

```
HOWARD_CALCULATION_REPORT="FindByPipelines,GenotypeConcordance,VAF_STATS,CALLING_QUALITY,CALLING_QUALI
TY_EXPLODE,VARTYPE,NOMEN"
```

Les filtres de priorisation utilisés pour classer les variants d'un VCF crées un classement par score des variants ainsi qu'un flag de validation et des commentaires prédéfinis pour chaque filtre. De la même manière que pour les paramètres d'annotation, ces paramètres sont utilisés par les règles d'annotation « howard » et « howard_minimal » ainsi que pour les rapports. Un paramètre de filtre de priorisation par défaut est définit dans le fichier « env_header.sh » (`$HOWARD_PRIORITIZATION_DEFAULT`)

Filtre de priorisation de l'annotateur « howard » :

```
HOWARD_PRIORITIZATION="default"
```

Filtre de priorisation de l'annotateur « howard_minimal » :

```
HOWARD_PRIORITIZATION_MINIMAL="default"
```

Filtre de priorisation des VCF du rapport final :

```
HOWARD_PRIORITIZATION_REPORT="default"
```

Chaque VCF est convertit au format TSV pour une visualisation plus adaptée dans un tableur. La liste des colonnes a intégrer dans ces fichiers peut être définie (ALL pour lister toutes les autres annotations non spécifiquement listées), ainsi que l'ordre des variants en fonction de deux champs :

```
HOWARD_FIELDS="NOMEN,PZFlag,PZScore,PZComment,CNOMEN,PNOMEN,location,outcome,VAF_average,dbSNP,dbSNPNo
nFlagged,popfreq,snpeff_impact,ALL"
HOWARD_SORT_BY="PZFlag,PZScore"
HOWARD_ORDER_BY="DESC,DESC"
```

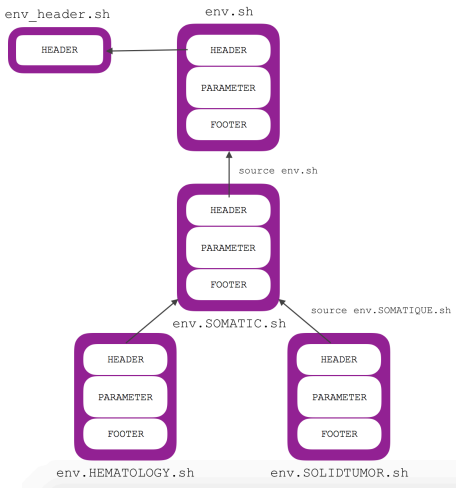
### F. Vérification des paramètres

le fichier env_footer.sh vérifie les paramètres saisis précédemment

```
source $STARK_FOLDER/env_footer.sh
```

## HÉRITAGE DES APPLICATIONS

La configuration par défaut de STARK est définie dans le fichier env.sh. Pour configurer une application sans devoir redéfinir tout les paramètres, il est possible de faire hériter les applications entres elles. La structure d'une application est identique à celle de env.sh (HEADER - PARAMETER - FOOTER).




*Exemple:*

L'application par défaut (env.sh) utilise le header principal (env_header.sh) . Une application B (env.B.sh) hérite d'une autre application A (env.A.sh) en définissant le header de l'application B comme le fichier de configuration de l'application A:

```
source env.A.sh
```

L'application SOMATIQUE (env.SOMATIC.sh) hérite de l'application par défaut (env.sh), tout en définissant ses paramètres spécifiques qui divergent de l'application par défaut. De la même manière les applications HEMATOLOGIE et TUMEUR SOLIDE héritent de l'application SOMATIQUE.

## LISTE DES ALIGNEURS-CALLERS-ANNOTATEURS

Un certain nombre d'outils sont déjà disponibles dans STARK permettant ainsi de créer ses propres pipelines.

*Exemples de listes d'aligners, callers, annotateurs disponibles :*

| STEP TYPE | STEP NAME | STEP DESCRIPTION |
|---|---|---|
| ALIGNER | bowtie | BOWTIE - Bowtie alignment algorithm |
| | bwaaln | BWA ALN - First BWA algorithm |
| | bwamem | Last powerful algorithm. From FASTQ files. BAM_CHECK_STEP switched off |
| | bwamem_FromFASTQ | BWA MEM - Last powerful algorithm from FASTQ files. Please switch off BAM_CHECK_STEP |
| | bwamem_FromUBAM | BWA MEM - Last powerful algorithm from unaligned BAM files |
| | bwasw | BWA SW - Smith Watermann algorithm |
| ANNOTATOR | howard | HOWARD annotates and prioritizes variants |
| | snpeff | snpEff annotation |
| CALLER | VarScan | SAMTOOLS/VARSCAN - by default |
| | VarScan_EXOME_SOMATIC | SAMTOOLS/VARSCAN - design for SOMATIC mutation on WES, VAF>0.05 ALT>5 DP>50 p- |
| | VarScan_HEMATOLOGY | SAMTOOLS/VARSCAN - design for HEMATOLOGY mutation, VAF>0.03 ALT>4 DP>50 p- |
| | VarScan_SOLIDTUMOR | SAMTOOLS/VARSCAN - design for SOLIDTUMOR mutation, VAF>0.01 ALT>5 DP>30 p- |
| | VarScan_SOMATIC | SAMTOOLS/VARSCAN - design for SOMATIC mutation, VAF>0.01 ALT>4 DP>50 p-value<1e-1 |
| | gatkHC | GATK Haplotype Caller - by default |
| | gatkHC_EXOME | GATK Haplotype Caller - designed for EXOME analysis |
| | gatkHC_EXOME_SOMATIC | GATK Haplotype Caller - designed for EXOME_SOMATIC analysis |
| | gatkHC_GENOME | GATK Haplotype Caller - designed for GENOME analysis |

| STEP TYPE | STEP NAME | STEP DESCRIPTION |
|---|---|---|
| | gatkHC_GERMLINE | GATK Haplotype Caller - designed for GERMLINE discovery |
| | gatkHC_HEMATOLOGY | GATK Haplotype Caller - designed for HEMATOLOGY variant discovery |
| | gatkHC_SOLIDTUMOR | GATK Haplotype Caller - designed for SOLIDTUMOR variant discovery |
| | gatkUG | GATK Unified Genotyper - by default |
| | gatkUG_EXOME | GATK Unified Genotyper - designed for EXOME discovery |
| | gatkUG_EXOME_SOMATIC | GATK Unified Genotyper - designed for EXOME_SOMATIC discovery |
| | gatkUG_GENOME | GATK Unified Genotyper - designed for GENOME discovery |
| | gatkUG_GERMLINE | GATK Unified Genotyper - designed for GERMLINE discovery |
| | gatkUG_HEMATOLOGY | GATK Unified Genotyper - designed for HEMATOLOGY variant discovery |
| | gatkUG_SOLIDTUMOR | GATK Unified Genotyper - designed for SOLIDTUMOR variant discovery |
| | itdseek | ITDSeek - FLT3 ITD detection |
| | samtools | SAMTOOLS/BCLTOOLS - by default, no filtering |
| | samtools_DP100 | SAMTOOLS/BCLTOOLS - filtring variant depth<100 |
| | samtools_DP10 | SAMTOOLS/BCLTOOLS - filtring variant depth<10 |
| | samtools_DP30 | SAMTOOLS/BCLTOOLS - filtring variant depth<30 |
| | canoes | CANOES - CNV caller |
| POST_ALIGNMENT | clipping | Prime Clipping for Amplicon sequencing |
| | compress | BAM compression |
| | markduplicates | Mark duplicated reads in BAM with PICARD MarkDuplicates |
| | markduplicatesSAMTOOLS | Mark duplicated reads in BAM. With SAMTOOLS rmdup |
| | realignment | Local Realignment of reads in BAM |
| | recalibration | BaseRecalibrator of reads in BAM. Warning |
| | sorting | BAM sorting |

# INSTALLATION AVANCÉE

Si vous faites le choix d'une installation complète de STARK (sans VM ni docker), il est nécessaire d'installer tous les outils et bases de données utilisés par STARK individuellement.

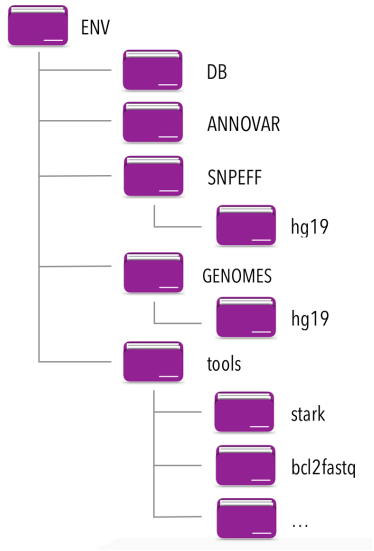
## INSTALLATION DES OUTILS SYSTÈME

Les outils suivants sont à installer sur la machine.

*La listes des outils et des versions*

| Prérequis | Version | Description |
|---|---|---|
| make | 4.1 | A build automation tool that automatically builds executable programs and libraries from source code. |
| bc | 1.06.95 | A language that supports arbitrary precision numbers with interactive execution of statements |
| wget | 1.12 | retrieves content from web servers |
| ghostscript | 9.0 | an interpreter for the PostScript language an for PDF |
| libgomp | 4.0 | An offloading and multiprocessing runtime library |
| java | 1.8.0_65 | A high-level programming language developed by Sun Microsystems |
| rsync | 3.0.6 | efficiently transferring and synchronizing files across computer systems |

## INSTALLATION DES OUTILS ET BASES DE DONNÉES




L'ensemble des outils et bases de données sont à installer dans un répertoire TOOLS. Il contiendra l'outil STARK et l'ensemble des outils nécessaires à sont bon fonctionnement. L'arborescence doit respecter l'architecture Galaxy (TOOL/RELEASE/BIN).

### Installation de STARK

S'adresser à Contact pour obtenir l'outil STARK (à déposer dans le dossier tools).

À ce stade, l'arborescence devrait être la suivante:

```
<INSTALLATION>/<ENV>/tools/stark/0.9.16b/bin/STARK.sh
```

### Installation des outils utilisés par STARK

Une fois téléchargé, il est impératif de déposer les outils ci-après dans le dossier tools :

```
<INSTALLATION>/<ENV>/tools
```

Tout les outils utilisés par STARK sont définis dans le fichier suivant :

```
<INSTALLATION>/ENV/tools/stark/current/bin/env_tools.sh
```

Chaque outil est défini de la manière suivante (Il est impératif que chaque outil soit redéfini selon votre choix d'installation).

```
OUTIL="Le nom de l'outil"
OUTIL_VERSION="La version de l'outil"
OUTIL_DESCRIPTION"La description de l'outil"
OUTIL_REF="La référence de l'outil (si elle existe)"
```

*Liste des outils et des versions*

| Prérequis | Version | Description |
|---|---|---|
| AnnotSV | 1.0 | |
| annovar | 2015Mar22 | An efficient software tool to utilize update-to-date information to functionally annotate genetic variants detected from diverse genomes |
| bcftools | 1.8 | Reading/writing BCF2/VCF/gVCF files and calling/filtering/summarising SNP and short indel sequence variants |
| bcl2fastq | 2.17.1.14 | BCL to FASTQ conversion |
| bedtools | 2.17.0<br>2.25.0 | A powerful toolset for genome arithmetic |
| bowtie2 | 2.2.8 | Bowtie 2 is an ultrafast and memory-efficient tool for aligning sequencing reads to long reference sequences |
| bwa | 0.7.17 | Package for mapping low-divergent sequences against a large reference genome |
| Canoes | 1.0.0 | |
| fastqc | 0.11.7 | A quality control tool for high throughput sequence data |
| gatk | 3.8-0 | The toolkit offers a wide variety of tools, with a primary focus on variant discovery and genotyping as well as strong emphasis on data quality assurance |
| igvtools | 2.3.98 | Provides a set of tools for pre-processing data files |
| java | 1.8.0_65 | |
| mutect | 1.1.4 | A method developed at the Broad Institute for the reliable and accurate identification of somatic point mutations in next generation sequencing data of cancer genomes |
| picard | 2.18.5 | Java command line tools for manipulating high-throughput sequencing data (HTS) data and format |
| R | 3.2.2 | A Language and Environment for Statistical Computing |
| samtools | 1.8 | Reading/writing/editing/indexing/viewing SAM/BAM/CRAM format |
| snpeff | 4.3t | Genetic variant annotation and effect prediction toolbox. It annotates and predicts the effects of variants on genes (such as amino acid changes) |
| tabix | 1.8 | Indexing VCF files |
| varscan | v2.3.9 | Variant detection in massively parallel sequencing data |
| vcftools | 0.1.13 | Provide easily accessible methods for working with complex genetic variation data in the form of VCF files |

Pour obtenir les outils développés en interne, s'adresser à Contact. Une fois les outils obtenus, ils sont à déposer les dans le dossier tools.

*La liste des outils développés en interne*

| Outil | Version | Description |
|---|---|---|
| FATBAM | 0.9.9b | Clipping Amplicons' Primers |
| HOWARD | 0.9.13b | Highly Open and Valuable tool for Variant Annotation & Ranking |
| COULSON | 0.9b | Ajoute des commandes (analyses) en file d'attente et elles sont exécutés une par une sans problème de surcharge. Plusieurs options sont disponibles. |
| DAEMON | 0.9b | L'outil Daemon permet d'automatiser le lancement des analyses. |

Les bases de données dans la liste ci-dessous sont à installer dans les répertoire TOOLS. Une TARBall contenant toutes les DBs utilisées par STARK est disponible (cf. Contact). Si les DBs ne sont pas installées, HOWARD les télécharges automatiquement.

*la listes des DBs et des versions*

| BD | Description |
|---|---|
| `Snpeff` | DB snpeff version 4.3t |
| `Annovar` | DB annovar version 2015Mar22 |
| `DB` | DB minimum: RefSeq.hg19.bed, dbsnp_138.hg19.vcf.gz, dbsnp_138.hg19.vcf.gz.tbi |
| `genomes` | Genome minimum: GRCH37 (fichier hg19.fa) |

## CRÉATION D'UN NOUVEAU DESIGN

Cette étape permet d'intégrer un nouveau design (panel de gènes, exome…) à partir du fichier bed des zones couvertes définies par le fournisseur du kit de séquençage. Les fichiers manifest (ou bed) et genes sont essentiels pour les calculs de couvertures, le fichier transcripts permet de définir les transcrits de chaque gène à utiliser par défaut. Seul le fichier manifest est conseillé (sauf pour une analyse sur l'ensemble du génome), tous les autres fichiers sont automatiquement générés à partir de celui-ci.

Il est conseillé de respecter la mise en forme suivante :

```
<INSTALLATION>/<MANIFESTS>/[application].[librarie].[version].manifest
<INSTALLATION>/<MANIFESTS>/[application].[librarie].[version].manifest.bed
<INSTALLATION>/<MANIFESTS>/[application].[librarie].[version].manifest.genes
<INSTALLATION>/<MANIFESTS>/[application].[librarie].[version].manifest.transcripts
```

Exemple de mise en forme :

```
MYOPATHIE.QXT.V1.manifest
```

| Champ | Description |
|---|---|
| pathologie | l'application utilisée. p.e. WES dans le cas d'une exome, PATHOLOGIE dans le cas d'un panel de gène. |
| librarie | Kit de préparation de la librairie |
| version | version du design |

### FICHIER MANIFEST

Le fichier manifest définit les régions séquencées correspondant aux reads présents dans les données brutes (fastq, bam…).

Le fichier manifest peut être demandé au fournisseur du kit de séquençage ou bien être créé à partir d'un fichier bed. Seul un fichier manifest permet de définir les primers dans le cas de la technologie amplicon, contrairement au fichier bed.

Exemple d'un fichier manifest :

```
[Header]
ReferenceGenome        Homo_sapiens\UCSC\hg19\Sequence\WholeGenomeFASTA
[Regions]
Name        Chromosome Amplicon Start Amplicon End   Upstream Probe Length  Downstream Probe Length
EGFR_ex19   chr7       55242379       55242574       30      30
EGFR_ex18   chr7       55241584       55241796       29      30
EGFR_ex21   chr7       55259375       55259589       30      21
EGFR_ex20   chr7       55248957       55249194       28      22
```

Génération d'un fichier manifest à partir d'un fichier bed

```
echo "[Header]" > my.manifest
echo "ReferenceGenome  Homo_sapiens\UCSC\hg19\Sequence\WholeGenomeFASTA >> my.manifest
echo "[Regions] >> my.manifest
echo "Name  Chromosome Amplicon Start Amplicon End" >> my.manifest
awk '{print $1_$2_$3"\t"$1"\t"$2"\t"$3}' my.bed >> my.manifest
```

### FICHIER BED

Le fichier bed définit les régions qui seront analysées par les pipelines, notamment en limitant l'étape de calling de variants.

Dans le cas où aucun fichier bed n'est pas fourni, le fichier bed est automatiquement généré à partir d'un fichier manifest.

Dans le cas d'une analyse par run, le fichier manifest avec l'extension « .bed » est automatiquement détecté (ex: pour un fichier manifest « designA.manifest », le fichier bed « designA.manifest.bed » sera utilisé s'il existe).

Exemple de fichier bed :

```
chr7        55241584   55241796       +       EGFR_ex18
chr7        55242379   55242574       +       EGFR_ex19
chr7        55248957   55249194       +       EGFR_ex20
chr7        55259375   55259589       +       EGFR_ex21
```

Génération d'un fichier bed à partir d'un fichier manifest

```
awk '{print $2"\t"$3"\t"$4}' my.manifeset >> my.bed
```

## FICHIER GENES

Le fichier genes défini précisément les régions correspondant à chaque gène. Ce fichier permet de calculer la couverture de chaque gènes, c'est à dire le pourcentage de bases séquencées pour une profondeur définie (option dans le fichier de configuration permettant de définir plusieurs profondeurs).

Dans le cas où aucun fichier genes n'est fourni, le fichier genes est automatiquement généré à partir du fichier manifest (ou bed) et de la définition des genes dans RefSeq.

Dans le cas d'une analyse par run, le fichier manifest avec l'extension « .genes » est automatiquement détecté (ex: pour un fichier manifest « designA.manifest », le fichier genes « designA.manifest.genes » sera utilisé s'il existe).

Exemple de fichier genes :

```
chr7        55242419   55242544       EGFR
chr7        55241613   55241766       EGFR
chr7        55259415   55259610       EGFR
chr7        55248985   55249216       EGFR
```

Génération d'un fichier genes à partir d'un fichier bed en utilisant l'annotation RefSeq (format bed) :

```
$BEDTOOLS/intersectBed -a refseq.bed -b my.bed -wa | awk '{print $1"\t"$2"\t"$3"\t"$5}' | uniq >
my.genes
```

Exemple de fichier d'annotation RefSeq au format bed :

```
chr7        55236215   55236328       +       EGFR    NM_201282,exon16
chr7        55237999   55238738       +       EGFR    NM_201284,exon16
chr7        55238867   55238906       +       EGFR    NM_005228,exon16
chr7        55240675   55240817       +       EGFR    NM_005228,exon17
chr7        55241613   55241736       +       EGFR    NM_005228,exon18
chr7        55242414   55242513       +       EGFR    NM_005228,exon19
chr7        55248985   55249171       +       EGFR    NM_005228,exon20
chr7        55259411   55259567       +       EGFR    NM_005228,exon21
chr7        55260458   55260534       +       EGFR    NM_005228,exon22
chr7        55266409   55266556       +       EGFR    NM_005228,exon23
```

## FICHIER TRANSCRIPTS

Le fichier transcripts permet de définir les transcrits (avec ou sans version) de chaque gène à utiliser par défaut. Dans le cas de multiple transcrits par défaut, le transcrit prioritaire est celui arrivant en premier dans la liste (dans le cas de l'exemple ci-dessous, le transcrit « transcipt2a » est prioritaire au transcript « transcript2b » pour le gène « gene2 »).

Dans le cas d'une analyse par run, le fichier manifest avec l'extension « .transcripts » est automatiquement détecté (ex: pour un fichier manifest « designA.manifest », le fichier transcripts « designA.manifest.transcripts » sera utilisé s'il existe).

Exemple de fichier transcript :

```
transcript1    gene1
transcript2a   gene2
transcript2b   gene2
transcript3    gene3
```

## CONTACT


### PLATEFORME BIOINFORMATIQUE

Hôpitaux Universitaires de Strasbourg (UF7363)

1, place de l'hôpital, 67091 Strasbourg

+33 3 88 12 75 38

bioinfo@chru-strasbourg.fr

## HISTORIQUE DE RÉVISION

| Date de révision | Motif de la révision |
|---|---|
| 17-11-2017 | Création du Document |
| 15-06-2018 | Mise à jour pour la version 0.9.16b |
| 09-10-2018 | Mise à jour pour la version 0.9.17b |