Cancer Workflow Sprint Review

Refactoring for version 0.9

Maxime Garcia

Barntumörbanken / SciLifeLab

2016/11/14

What is working without problem

Currently working on version 0.9

- Preprocessing (was already working)
- Starting from bam files (so they are aligned as a T/N pair)
- Easy configuration with profiles
- Easy execution
- MuTect1 (concatenation of results refactored)
- MuTect2 (concatenation of results refactored)
- Strelka (was already working)
- VarDict (concatenation of results refactored)
- New option to run CAW with a specific UPPMAX project number

What is working without problem

Currently working on version 0.9

- Preprocessing (was already working)
- Starting from bam files (so they are aligned as a T/N pair)
- Easy configuration with profiles
- Easy execution
- MuTect1 (concatenation of results refactored)
- MuTect2 (concatenation of results refactored)
- Strelka (was already working)
- VarDict (concatenation of results refactored)
- New option to run CAW with a specific UPPMAX project number

What is working without problem

Currently working on version 0.9

- Preprocessing (was already working)
- Starting from bam files (so they are aligned as a T/N pair)
- Easy configuration with profiles
- Easy execution
- MuTect1 (concatenation of results refactored)
- MuTect2 (concatenation of results refactored)
- Strelka (was already working)
- VarDict (concatenation of results refactored)
- New option to run CAW with a specific UPPMAX project number

What is working without problem

Currently working on version 0.9

- Preprocessing (was already working)
- Starting from bam files (so they are aligned as a T/N pair)
- Easy configuration with profiles
- Easy execution
- MuTect1 (concatenation of results refactored)
- MuTect2 (concatenation of results refactored)
- Strelka (was already working)
- VarDict (concatenation of results refactored)
- New option to run CAW with a specific UPPMAX project number

What is working without problem

Currently working on version 0.9

- Preprocessing (was already working)
- Starting from bam files (so they are aligned as a T/N pair)
- Easy configuration with profiles
- Easy execution
- MuTect1 (concatenation of results refactored)
- MuTect2 (concatenation of results refactored)
- Strelka (was already working)
- VarDict (concatenation of results refactored)
- New option to run CAW with a specific UPPMAX project number

What is working without problem

Currently working on version 0.9

- Preprocessing (was already working)
- Starting from bam files (so they are aligned as a T/N pair)
- Easy configuration with profiles
- Easy execution
- MuTect1 (concatenation of results refactored)
- MuTect2 (concatenation of results refactored)
- Strelka (was already working)
- VarDict (concatenation of results refactored)
- New option to run CAW with a specific UPPMAX project number

What is working without problem

Currently working on version 0.9

- Preprocessing (was already working)
- Starting from bam files (so they are aligned as a T/N pair)
- Easy configuration with profiles
- Easy execution
- MuTect1 (concatenation of results refactored)
- MuTect2 (concatenation of results refactored)
- Strelka (was already working)
- VarDict (concatenation of results refactored)
- New option to run CAW with a specific UPPMAX project number

What is working without problem

Currently working on version 0.9

- Preprocessing (was already working)
- Starting from bam files (so they are aligned as a T/N pair)
- Easy configuration with profiles
- Easy execution
- MuTect1 (concatenation of results refactored)
- MuTect2 (concatenation of results refactored)
- Strelka (was already working)
- VarDict (concatenation of results refactored)
- New option to run CAW with a specific UPPMAX project number

What is working without problem

Currently working on version 0.9

- Preprocessing (was already working)
- Starting from bam files (so they are aligned as a T/N pair)
- Easy configuration with profiles
- Easy execution
- MuTect1 (concatenation of results refactored)
- MuTect2 (concatenation of results refactored)
- Strelka (was already working)
- VarDict (concatenation of results refactored)
- New option to run CAW with a specific UPPMAX project number

Easier profile configuration

nextflow.config

```
profiles {
standard {
includeConfig 'config/standard.config'
includeConfig 'config/milou.config'
}
}
```

milou.config

```
process {
//MILOU SPECIFIC DETAILS (QUEUE, EXECUTOR, MEMORY, CPUS...)
executor = 'slurm'
memory = '128.GB'

//ERROR HANDLING AND RESTARTING STRATEGY
errorStrategy = task.exitStatus == 143 ? 'retry' : 'terminate'
maxRetries = 3
maxErrors = '-1'

//MODULE FOR EACH PROCESS
$MapReads.module = ['bioinfo-tools', 'bwa/0.7.13', 'samtools/1.3']
}

params { //MILOU SPECIFIC PATHS
genome        = 'path_to_refs_on_milou/human_g1k_v37_decoy.fasta'
genomeIndex   = 'path_to_refs_on_milou/human_g1k_v37_decoy.fasta.fai'
mutect1Home   = '/sw/apps/bioinfo/mutect/1.1.5/milou'
gatkHome      = '/sw/apps/bioinfo/GATK/3.6'
vardictHome   = '/sw/apps/bioinfo/VarDictJava/1.4.5/milou/VarDictJava'
}
```

Easier profile configuration

nextflow.config

```
profiles {
standard {
includeConfig 'config/standard.config'
includeConfig 'config/milou.config'
}
}
```

milou.config

```
process {
//MILOU SPECIFIC DETAILS (QUEUE, EXECUTOR, MEMORY, CPUS...)
executor = 'slurm'
memory = '128.GB'

//ERROR HANDLING AND RESTARTING STRATEGY
errorStrategy = task.exitStatus == 143 ? 'retry' : 'terminate'
maxRetries = 3
maxErrors = '-1'

//MODULE FOR EACH PROCESS
$MapReads.module = ['bioinfo-tools', 'bwa/0.7.13', 'samtools/1.3']
}

params { //MILOU SPECIFIC PATHS
genome        = 'path_to_refs_on_milou/human_g1k_v37_decoy.fasta'
genomeIndex   = 'path_to_refs_on_milou/human_g1k_v37_decoy.fasta.fai'
mutect1Home   = '/sw/apps/bioinfo/mutect/1.1.5/milou'
gatkHome      = '/sw/apps/bioinfo/GATK/3.6'
vardictHome   = '/sw/apps/bioinfo/VarDictJava/1.4.5/milou/VarDictJava'
}
```

Easy execution

Execute on milou from github

```
nextflow run SciLifeLab/CAW --sample mysample.tsv --steps
preprocessing,MuTect1,MuTect2,Strelka,VarDict --project b2015110
```

Execute on milou with cloned or downloaded repo

```
nextflow run main.nf --sample mysample.tsv --steps
preprocessing,MuTect1,MuTect2,Strelka,VarDict --project b2015110
```

Update Workflow

```
nextflow pull SciLifeLab/CAW // git pull
nextflow run SciLifeLab/CAW --sample mysample.tsv --steps
preprocessing,MuTect1,MuTect2,Strelka,VarDict --project b2015110 -resume
```

Control which version is used

```
nextflow run SciLifeLab/CAW --sample mysample.tsv --steps
preprocessing,MuTect1,MuTect2,Strelka,VarDict --project b2015110 -r v0.9
```

Easy execution

Execute on milou from github

```
nextflow run SciLifeLab/CAW --sample mysample.tsv --steps
preprocessing,MuTect1,MuTect2,Strelka,VarDict --project b2015110
```

Execute on milou with cloned or downloaded repo

```
nextflow run main.nf --sample mysample.tsv --steps
preprocessing,MuTect1,MuTect2,Strelka,VarDict --project b2015110
```

Update Workflow

```
nextflow pull SciLifeLab/CAW // git pull
nextflow run SciLifeLab/CAW --sample mysample.tsv --steps
preprocessing,MuTect1,MuTect2,Strelka,VarDict --project b2015110 -resume
```

Control which version is used

```
nextflow run SciLifeLab/CAW --sample mysample.tsv --steps
preprocessing,MuTect1,MuTect2,Strelka,VarDict --project b2015110 -r v0.9
```

Easy execution

Execute on milou from github

```
nextflow run SciLifeLab/CAW --sample mysample.tsv --steps
preprocessing,MuTect1,MuTect2,Strelka,VarDict --project b2015110
```

Execute on milou with cloned or downloaded repo

```
nextflow run main.nf --sample mysample.tsv --steps
preprocessing,MuTect1,MuTect2,Strelka,VarDict --project b2015110
```

Update Workflow

```
nextflow pull SciLifeLab/CAW // git pull
nextflow run SciLifeLab/CAW --sample mysample.tsv --steps
preprocessing,MuTect1,MuTect2,Strelka,VarDict --project b2015110 -resume
```

Control which version is used

```
nextflow run SciLifeLab/CAW --sample mysample.tsv --steps
preprocessing,MuTect1,MuTect2,Strelka,VarDict --project b2015110 -r v0.9
```

Easy execution

Execute on milou from github

```
nextflow run SciLifeLab/CAW --sample mysample.tsv --steps
preprocessing,MuTect1,MuTect2,Strelka,VarDict --project b2015110
```

Execute on milou with cloned or downloaded repo

```
nextflow run main.nf --sample mysample.tsv --steps
preprocessing,MuTect1,MuTect2,Strelka,VarDict --project b2015110
```

Update Workflow

```
nextflow pull SciLifeLab/CAW // git pull
nextflow run SciLifeLab/CAW --sample mysample.tsv --steps
preprocessing,MuTect1,MuTect2,Strelka,VarDict --project b2015110 -resume
```

Control which version is used

```
nextflow run SciLifeLab/CAW --sample mysample.tsv --steps
preprocessing,MuTect1,MuTect2,Strelka,VarDict --project b2015110 -r v0.9
```

Whole code refactoring

- Add tag for each process (better processes tracking)
- Remove loading modules and error strategies (code reading simplified)
- Remove unused old functions (code simplified)
- Add conditionnal process management (better control of steps)
- Constant effort on refactoring to have code more accessible and easier to maintain
- Better help, and mutualization of help with other developpers using Nextflow at SciLifeLab

Whole code refactoring

- Add tag for each process (better processes tracking)
- Remove loading modules and error strategies (code reading simplified)
- Remove unused old functions (code simplified)
- Add conditionnal process management (better control of steps)
- Constant effort on refactoring to have code more accessible and easier to maintain
- Better help, and mutualization of help with other developpers using Nextflow at SciLifeLab

Whole code refactoring

- Add tag for each process (better processes tracking)
- Remove loading modules and error strategies (code reading simplified)
- Remove unused old functions (code simplified)
- Add conditionnal process management (better control of steps)
- Constant effort on refactoring to have code more accessible and easier to maintain
- Better help, and mutualization of help with other developpers using Nextflow at SciLifeLab

Whole code refactoring

- Add tag for each process (better processes tracking)
- Remove loading modules and error strategies (code reading simplified)
- Remove unused old functions (code simplified)
- Add conditionnal process management (better control of steps)
- Constant effort on refactoring to have code more accessible and easier to maintain
- Better help, and mutualization of help with other developpers using Nextflow at SciLifeLab

Whole code refactoring

- Add tag for each process (better processes tracking)
- Remove loading modules and error strategies (code reading simplified)
- Remove unused old functions (code simplified)
- Add conditionnal process management (better control of steps)
- Constant effort on refactoring to have code more accessible and easier to maintain
- Better help, and mutualization of help with other developpers using Nextflow at SciLifeLab

Whole code refactoring

- Add tag for each process (better processes tracking)
- Remove loading modules and error strategies (code reading simplified)
- Remove unused old functions (code simplified)
- Add conditionnal process management (better control of steps)
- Constant effort on refactoring to have code more accessible and easier to maintain
- Better help, and mutualization of help with other developpers using Nextflow at SciLifeLab

Great changes on Variant Calling

- Refactored Variant Calling organisation
- Better organisation and manipulation of Channels
- Removal of functions that were not used anymore
- Merged processes for concatenating Variant Calling results (MuTect1, MuTect2)
- Works also with VarDict and HaplotypeCaller results
- Only keeping final concatenated VCF

Great changes on Variant Calling

- Refactored Variant Calling organisation
- Better organisation and manipulation of Channels
- Removal of functions that were not used anymore
- Merged processes for concatenating Variant Calling results (MuTect1, MuTect2)
- Works also with VarDict and HaplotypeCaller results
- Only keeping final concatenated VCF

Great changes on Variant Calling

- Refactored Variant Calling organisation
- Better organisation and manipulation of Channels
- Removal of functions that were not used anymore
- Merged processes for concatenating Variant Calling results (MuTect1, MuTect2)
- Works also with VarDict and HaplotypeCaller results
- Only keeping final concatenated VCF

Great changes on Variant Calling

- Refactored Variant Calling organisation
- Better organisation and manipulation of Channels
- Removal of functions that were not used anymore
- Merged processes for concatenating Variant Calling results (MuTect1, MuTect2)
- Works also with VarDict and HaplotypeCaller results
- Only keeping final concatenated VCF

Great changes on Variant Calling

- Refactored Variant Calling organisation
- Better organisation and manipulation of Channels
- Removal of functions that were not used anymore
- Merged processes for concatenating Variant Calling results (MuTect1, MuTect2)
- Works also with VarDict and HaplotypeCaller results
- Only keeping final concatenated VCF

Great changes on Variant Calling

- Refactored Variant Calling organisation
- Better organisation and manipulation of Channels
- Removal of functions that were not used anymore
- Merged processes for concatenating Variant Calling results (MuTect1, MuTect2)
- Works also with VarDict and HaplotypeCaller results
- Only keeping final concatenated VCF