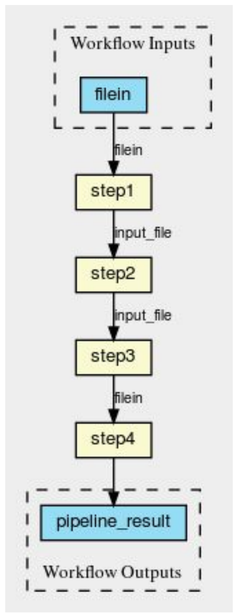
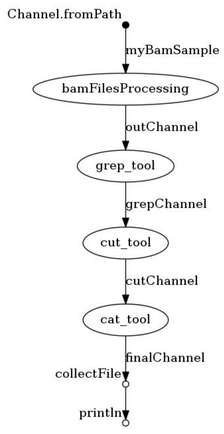
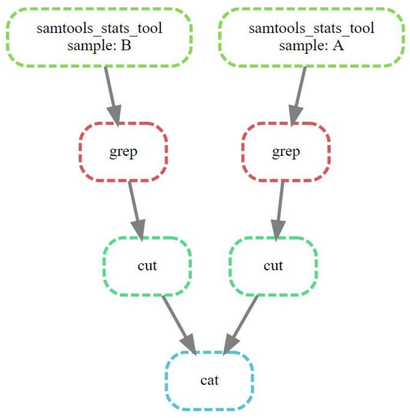
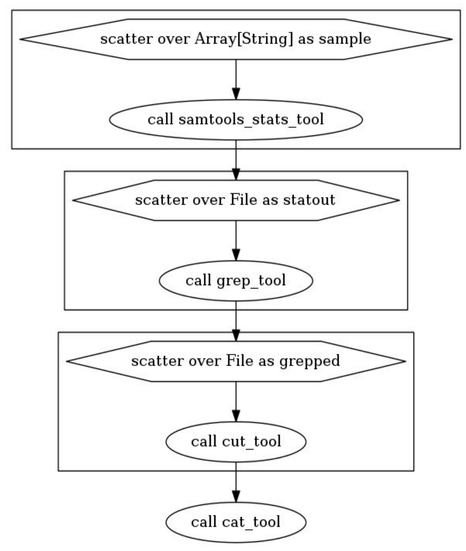
ISB-CGC

Comparison of Workflow Languages

	COMMON WORKFLOW LANGUAGE	nextflow	Snakemake	{wdl}
Installation & Dependencies	• Moderate to install • CWL runner	• Easy to install	• Moderate to install • Many dependencies to install and activate (bioconda, isolated environment…)	• Easy to install • Cromwell and WOMtool
Syntax	• Difficulty Level: Moderate high • Multiple-file workflow	• Difficulty Level: Moderate low • Well documented	• Difficulty Level: Low • Python friendly (also runs Python code)	• Difficulty Level: Moderate low • Java friendly
First Available	2014	2017	2012	2016
Other Info	• Lots of customizations • Interesting feature: validate	• Lots of customizations • Interesting features: watchPath, resume (not autoresume)	• Easier to handle inputs, output directories • Simple way to run (just type snakemake) • Can export to CWL	• Interesting features: WOMtool provides syntax validate, graphing, womgraph can also read CWL… • DOES NOT SUPPORT Directory type • Customized for GATK
Visualization	**Visualization via CWLviewer (web based and git link)** 	**Built-in visualization tool** 	**Built-in visualization tool** 	**Visualization via WOMtool** 
Sample Codes	*Below sample codes illustrate how one can accomplish the same process in different workflow languages*			

MyWorkflow.cwl

```
#!/usr/bin/env cwl-runner
cwlVersion: v1.0
class: Workflow
requirements:
  ScatterFeatureRequirement: {}
hints:
  DockerRequirement:
    dockerPull: chrishah/samtools:1.9
inputs:
  filein: File[]
outputs:
  pipeline_result:
    type: File
    outputSource: step4/catout
steps:
  step1:
    run: samtools_stats_tool.cwl
    scatter: [filein]
    scatterMethod: dotproduct
    in:
      filein: filein
    out:
      [statsout]

[...]
```

samtools_stats_tool.cwl

```
#samtools_stats_tool.cwl

cwlVersion: v1.0
class: CommandLineTool

baseCommand: [samtools, stats]

requirements:
  - class: InlineJavascriptRequirement

inputs:
  filein:
    type: File
    inputBinding:
      position: 1

outputs:
  statsout:
    type: File
    outputBinding:
      glob: "*.stats"

stdout: $(inputs.filein.path.split('/').pop() +\
'.stats')
```

MyWorkflow.nf

```
#!/usr/bin/env nextflow

myBamSample =\
Channel.fromPath('/home/thinh_vo/sample/*.bam')
params.outdir = 'Sam_results'

process bamFilesProcessing {
    cpus 1
    publishDir params.outdir, mode: 'copy'
    echo true
    container "gcr.io/genomics-tools/samtools"
    input:
    file query_file from myBamSample
    output:
    file "${query_file}_stats.txt" into\
outChannel
    """
    echo "processing ${query_file}"
    samtools stats ${query_file} >\
${query_file}_stats.txt
    """
}

[...]

finalChannel
      .collectFile(storeDir:'Sam_results')
      .println{ it.text }
```

Snakefile

```
IDS, = glob_wildcards("mapped_reads/{sample}.bam")
rule all:
    input:
        "final/final_gc_stats_out.txt"

rule samtools_stats_tool:
    input:
        "mapped_reads/{sample}.bam"
    output:
        "{sample}_gc_stats.txt"
    shell:
        "samtools stats {input} > {output}"\

[...]
```

MyWorkflow.wdl

```
workflow gcStats {
    File inputSamplesFile
    Array[Array[String]] inputSamples =\
read_tsv(inputSamplesFile)
    scatter (sample in inputSamples) {
     call samtools_stats_tool {
       input:
         filein=sample[0],
         filename=sample[1]
       }
     }

[...]

task samtools_stats_tool {
    File filein
    String filename
    runtime {
      docker: "gcr.io/genomics-tools/samtools"
    }
    command {
        samtools stats ${filein} >\
${filename}_gc_stats.txt
    }
    output {
         File statsout = "${filename}_gc_stats.txt"
    }
}

[...]
```

© 2020, ISB