



Medical Genomics Practical #1: Building reproducible workflows

International Agency for Research on Cancer Lyon, France

Emilie Mathian & Nicolas Alcala
Genetics section

23rd of November 2020

International Agency for Research on Cancer

Plan

Part I. Practicals: generating a multi-omic Tumor Map of rare lung tumors

- **Concepts:** Lung Neuroendocrine Neoplasms
- **Data**

Part II. Introduction to domain-specific languages for bioinformatics

- **Concepts:** Reproducibility, scalability, portability
- **Exercises**

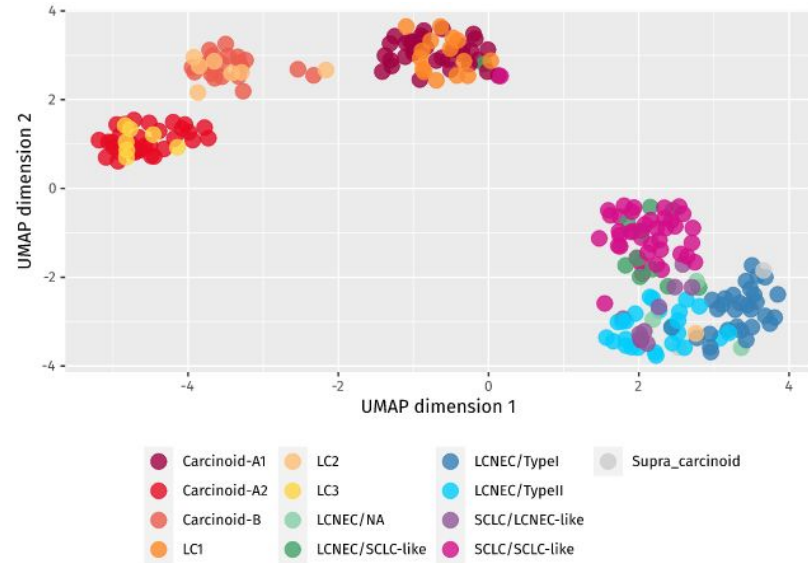
Part III. Practical: coding a simple RNA-seq pre-processing workflow

- **Concepts:** channels and processes
- **Exercises**

Part I. Medical Genomics practicals | *General goal*

Generating a multi-omic Tumor Map of rare lung tumors

- **Lung Neuroendocrine Neoplasms (LNENs)** are rare solid cancers originating from pulmonary neuroendocrine cells
- They are classified (WHO) into grade 1 (typical carcinoids), grade 2 (atypical carcinoids), and grade 3 (carcinoma)
- **Multi-omic datasets** (exomes, RNA-seq, and methylation arrays) were recently generated
- **Clinically relevant molecular groups** with different prognosis (from 88% to 30% 10-year survival) and potential therapeutic targets

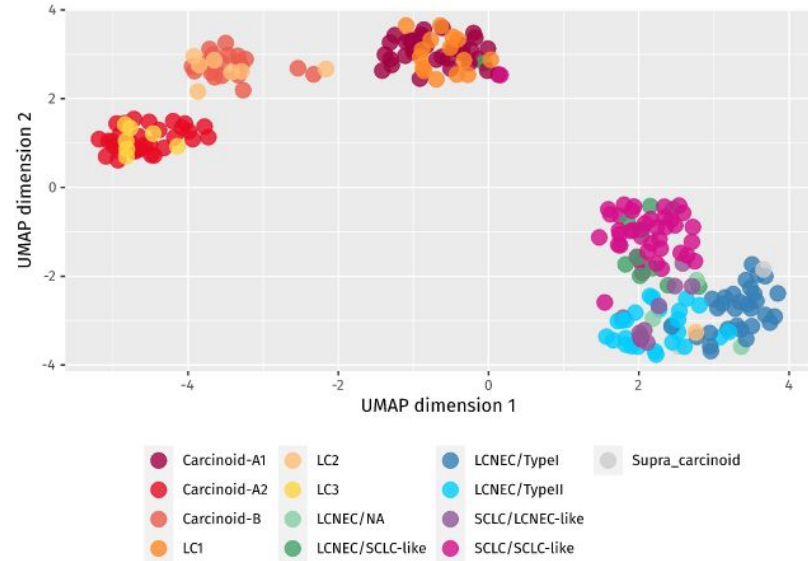


Tumor Map (UMAP) of lung neuroendocrine neoplasms. Source: <https://nextjournal.com/rarecancersgenomics/a-molecular-map-of-lung-neuroendocrine-neoplasms/> (Gabriel AAG*, Mathian E*, et al. GigaScience 2020). *Equally contributing.

Part I. Medical Genomics practicals | *General goal*

Generating a multi-omic Tumor Map of rare lung tumors

- **Data:** Laddha et al. (2019): 30 RNA-seq and 18 methylation arrays (GEO GSE118133; <https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE118133>)
- **Week 1:** build a workflow to pre-process the RNA-seq data and obtain a gene expression matrix
- **Week 2:** generate a “tumor map” from the gene expression and gene methylation matrices, analyze the results

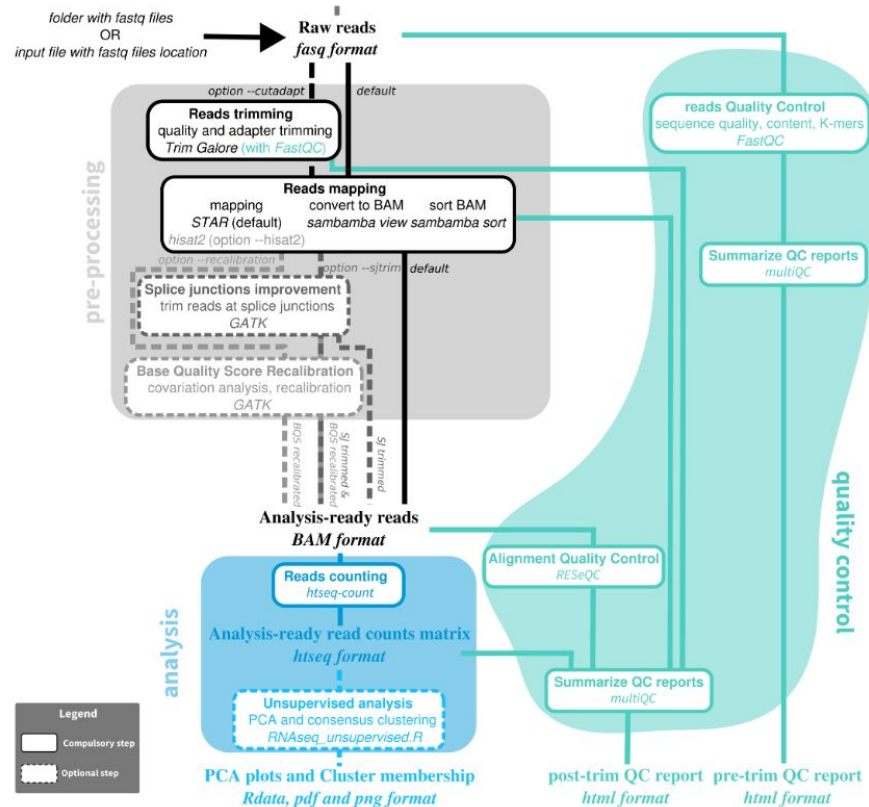


Tumor Map (UMAP) of lung neuroendocrine neoplasms. Source: <https://nextjournal.com/rarecancersgenomics/a-molecular-map-of-lung-neuroendocrine-neoplasms/> (Gabriel AAG*, Mathian E*, et al. GigaScience 2020). *Equally contributing.

Part II. Introduction to DSL for bioinformatics | *Example*

RNA-sequencing processing ([IARCbioinfo/RNAseq-nf](#))

- Multiple tasks have to be performed in succession (e.g., trimming, mapping, quantification) or in parallel (on each sample)
- Each task uses its own softwares and libraries, and can have incompatible requirements (e.g., python versions)



Part II. Introduction to DSL for bioinformatics | *Concepts*

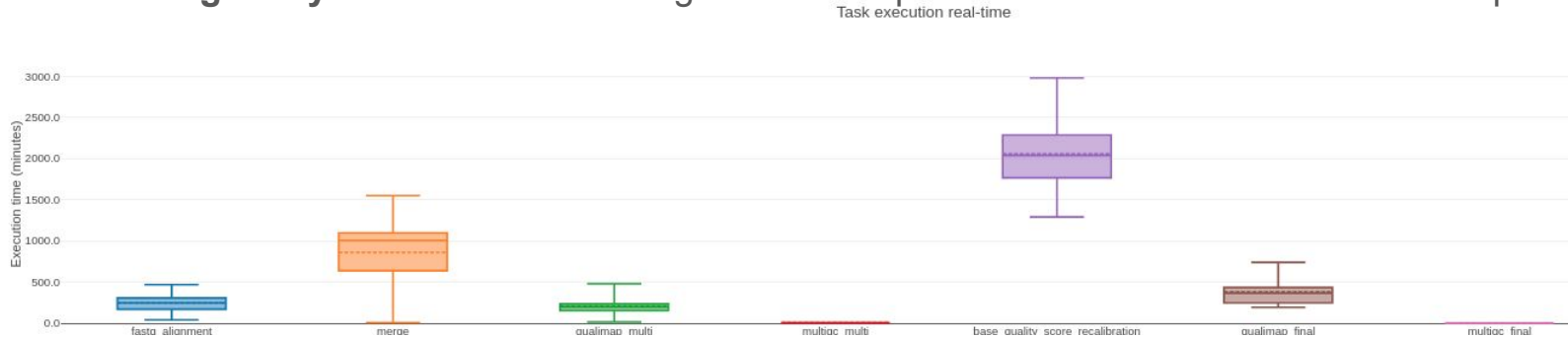
Workflow requirements for large-scale medical genomic projects

Reproducible: clinical and research applications need to be entirely reproducible

Scalable: easily run on large High-Performance Computing facilities

Portable: can run on various infrastructures (different OS, cloud)

Handle heterogeneity: works with conflicting software dependencies and various resource requirements



Example of duration of pre-processing. Tumor/normal pairs whole-genome sequencing (30X and 90X, respectively) for 10 patients, processed using workflow IARCbioinfo/alignment-nf (bwa+post-alignment+GATK BQSR+ pre- and post-alignment QC). **Total of 28,862.4 CPU hours (3 years and 3.5 months), actually processed in 3 days and 18 hrs.**

International Agency for Research on Cancer

Part II. Introduction to DSL for bioinformatics | *Concepts*

Solutions: domain-specific languages for bioinformatics

High-level programming languages designed to build workflows

- **Nextflow**: based on groovy (java); made at Centre for Genomic Regulation (Barcelona) / Seqera labs
- Snakemake: based on python; made at Institute of Human Genetics (Essen)
- WDL: own language; made at BROAD institute (Boston)

Part II. Introduction to DSL for bioinformatics | *Concepts*

Nextflow

Used by groups all over the world

The nf-core community is spread all over the globe and includes a large number of contributing users.

[See a complete list »](#)

HIRI **HELMHOLTZ**
Institute for RNA-based Infection Research

  City of Hope   MRC Human Genetics Unit  NYU  GMI GREGOR MENDEL INSTITUTE OF MOLECULAR PLANT BIOLOGY 

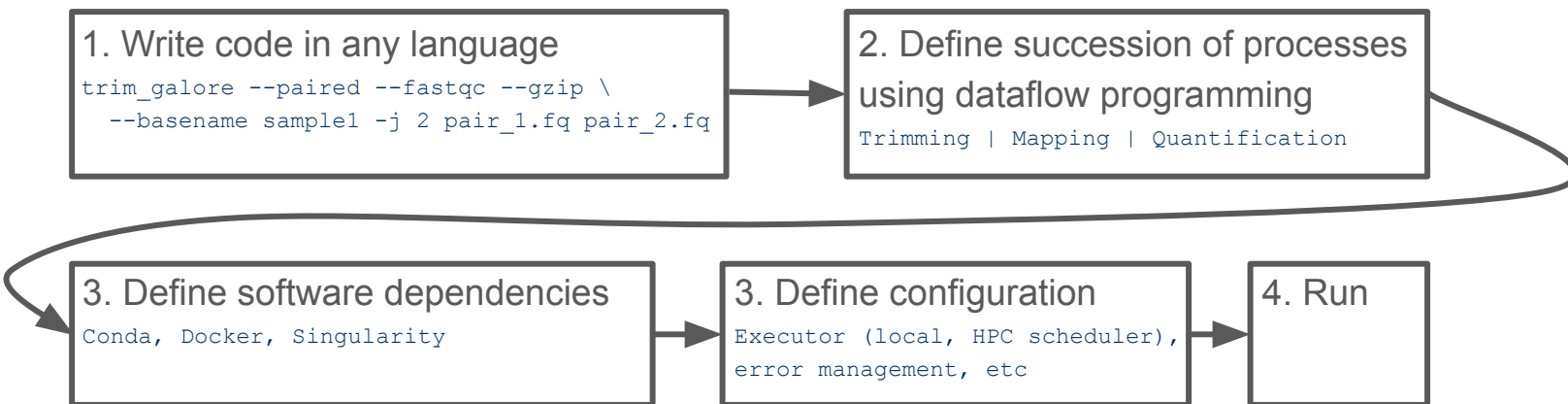
NBS  Boehringer Ingelheim  Institut Pasteur  University of Colorado Boulder

Berkeley  CHAN ZUCKERBERG BIOHUB  UCL CANCER INSTITUTE  IMP

International Agency for Research on Cancer

Part II. Introduction to DSL for bioinformatics | *Concepts*

Nextflow design



Part II. Introduction to DSL for bioinformatics | *Concepts*

Nextflow language features

Dataflow language with implicit parallelization and scheduling of tasks:

- the **same set of tasks** is applied to all input files
- tasks are **automatically executed in the proper order** given their inputs and outputs
- **automatically handles submission** of jobs to HPC scheduler

Part II. Introduction to DSL for bioinformatics | *Concepts*

Running a workflow

```
nextflow run IARCbioinfo/FastQC-nf -profile conda --input_folder FASTQ/
```

Workflow name: either the *path to a nextflow script*, a *path to a directory* containing a nextflow config file pointing to a nextflow script, or a *github repository* (here, <https://github.com/IARCbioinfo/fastqc-nf>)

Workflow options (“--”): e.g., input files location

Nextflow options (“-”): e.g., profiles defining location of conda recipes listing software dependencies

Part III. Coding an RNA-seq workflow | *Practice running nextflow*

Practical 1. Questions 1 and 2

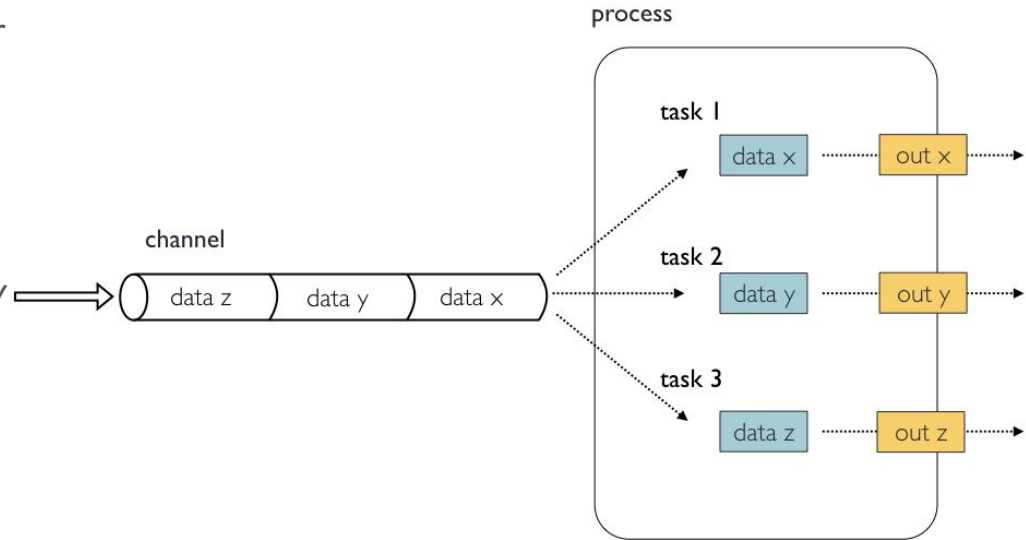
https://github.com/IARCbioinfo/medical_genomics_course/wiki/Practical-1

Don't forget to connect to kahoot for live questions (and a ranking)!
Link on slack

Part III. Coding an RNA-seq workflow | *Basics of Nextflow*

Coding a workflow

- **Channel:** array storing process inputs or outputs (e.g., paths to data)
- **Processes:** instructions to execute a task on an input channel element
 - *Note: Nextflow creates a work directory for each task, creates symbolic links to paths from Channel and executes the script defined in the process*



Part III. Coding an RNA-seq workflow | *Basics of Nextflow*

Coding a workflow

- **Channel:** array storing process inputs or outputs (e.g., paths to data)

Example: test.nf containing

```
TOTO = Channel.from("toto") //creates channel
                        .view()      //prints channel
```

```
> nextflow run test.nf
N E X T F L O W ~ version 20.10.0
```

```
Launching `test.nf` [loving_roentgen] -
revision: 9634ca93db
```

```
toto
```

Part III. Coding an RNA-seq workflow | *Basics of Nextflow*

Coding a workflow

- **Channel:** array storing process inputs or outputs (e.g., paths to data)
- **Channel factory:**
 - create a channel from a file path (note: can contain wildcards): `Channel.fromPath('bam/*.bam')`

```
> N E X T F L O W ~ version 20.10.0
Launching `test.nf` [deadly_yalow] - revision:
e4af11bd16
/home/nalcala/Medical_Genomics_TP1/data_test/B
AM/NA06984_N.bam
/home/nalcala/Medical_Genomics_TP1/data_test/B
AM/NA06984_T.bam
```


Part III. Coding an RNA-seq workflow | *Basics of Nextflow*

Coding a workflow

- **Channel:** array storing process inputs or outputs (e.g., paths to data)
- **Channel factory:**
 - create a channel from file pairs:
`Channel.fromFilePairs('bambai/*.{bam, bam.bai}')`

```
> N E X T F L O W ~ version 20.10.0
Launching `test.nf` [mighty_rubens] -
revision: 8e42d3ebe8
[NA06984_N,
[/home/nalcala/Medical_Genomics_TP1/data_test/
BAM/NA06984_N.bam,
/home/nalcala/Medical_Genomics_TP1/data_test/B
AM/NA06984_N.bam.bai]]
[NA06984_T,
[/home/nalcala/Medical_Genomics_TP1/data_test/
BAM/NA06984_T.bam,
/home/nalcala/Medical_Genomics_TP1/data_test/B
AM/NA06984_T.bam.bai]]
```

Part III. Coding an RNA-seq workflow | *Basics of Nextflow*

Coding a workflow

- **Channel:** array storing process inputs or outputs (e.g., paths to data)
- **Channel factory:**
 - create a channel directly from NCBI's small reads archive (SRA):
`Channel.fromSRA("SRP156394")`
 - see complete list of channel constructors at
<https://www.nextflow.io/docs/latest/channel.html#channel-factory>

```
> N E X T F L O W ~ version 20.10.0
Launching `test.nf` [sleepy_mandelbrot] -
revision: 8ef1436d72
[SRR7646258,
[/vol1/fastq/SRR764/008/SRR7646258/SRR7646258_
1.fastq.gz,
/vol1/fastq/SRR764/008/SRR7646258/SRR7646258_2
.fastq.gz]]
[SRR7646257,
[/vol1/fastq/SRR764/007/SRR7646257/SRR7646257_
1.fastq.gz,
/vol1/fastq/SRR764/007/SRR7646257/SRR7646257_2
.fastq.gz]]
...
```

Part III. Coding an RNA-seq workflow | *Basics of Nextflow*

Coding a workflow

- **Channel:** array storing process inputs or outputs (e.g., paths to data)
- **Channel factory:**
 - see complete list of channel constructors at <https://www.nextflow.io/docs/latest/channel.html#channel-factory>
- **Notes:**
 - *queue* Channels are consumed each time they are used
 - *singleton* Channels are reusable; created with `file()` and some other operators

```
ref = file("reference.fasta")
```

Part III. Coding an RNA-seq workflow | *Basics of Nextflow*

Coding a workflow

- **Processes:** instructions to execute a task on an input channel element

- **name**
- **input channel(s) name(s) and type(s)**
- **output channel(s) name(s) and type(s)**
- **(optional) path to output directory**
- **script (code to be executed)**

```
process test{  
  input:  
    path bam from bams  
  
  output:  
    path "${bam}.name" into names  
  
  publishDir "output"  
  
  shell:  
    '''  
    echo "!!{bam}" > !{bam}.name  
    '''  
}
```

Part III. Coding an RNA-seq workflow | *Basics of Nextflow*

Coding a workflow

- **Processes:** instructions to execute a task on an input channel element
 - **input channel(s) name(s) and type(s)**
 - **output channel(s) name(s) and type(s)**
 - types: *path* (absolute path of files), *val* (for variables, e.g., strings & numbers), *tuple* (multiple values of different types), *stdout* (everything printed by the script)
 - supports wildcards
 - “\$” to use nextflow variables
 - output paths are relative to the work folder

```
process test{  
    input:  
    path bam from bams  
  
    output:  
    path "${bam}.name" into names  
  
    publishDir "output"  
    shell:  
    '''  
    echo "!{bam}" > !{bam}.name  
    '''  
}
```

Part III. Coding an RNA-seq workflow | *Basics of Nextflow*

Coding a workflow

- **Processes:** instructions to execute a task on an input channel element

- (optional) path to output directory

By default, makes a *symbolic link* to the path of the actual output; can change behavior by specifying a “mode” (e.g., copy, move)

```
publishDir "output", mode: 'move'
```

Note: once moved, the data is inaccessible so needs to be done at the very end of the script

```
process test{
    input:
    path bam from bams

    output:
    path "${bam}.name" into names

    publishDir "output"
    shell:
    '''
    echo "${bam}" > ${bam}.name
    '''
}
```

Part III. Coding an RNA-seq workflow | *Basics of Nextflow*

Coding a workflow

- **Processes:** instructions to execute a task on an input channel element

- (optional) path to output directory

By default, makes a *symbolic link* to the path of the actual output; can change behavior by specifying a “mode” (e.g., copy, move)

```
publishDir "output", mode: 'move'
```

Note: *other directives include the memory usage or number of cpus to use (see list at <https://www.nextflow.io/docs/latest/process.html#directives>)*

```
process test{
    input:
    path bam from bams

    output:
    path "${bam}.name" into names

    publishDir "output"
    shell:
    '''
    echo "${bam}" > ${bam}.name
    '''
}
```


Part III. Coding an RNA-seq workflow | *Basics of Nextflow*

Coding a workflow

- **Processes:** instructions to execute a task on an input channel element
 - script (code to be executed)
 - “!” is used for nextflow variables (e.g., input variables) and “\$” for bash variables

```
process test{  
    input:  
    path bam from bams  
  
    output:  
    path "${bam}.name" into names  
  
    publishDir "output"  
    shell:  
    '''  
    echo "${bam}" > ${bam}.name  
    '''  
}
```

Part III. Coding an RNA-seq workflow | *Basics of Nextflow*

Coding a workflow

- **Nextflow scripting language:**
 - groovy syntax + nextflow-specific objects and operators
 - see more at <https://www.nextflow.io/docs/latest/script.html#script-page>

Part III. Coding an RNA-seq workflow | *Write a simple script*

Practical 1. Question 3

https://github.com/IARCbioinfo/medical_genomics_course/wiki/Practical-1

Part III. Coding an RNA-seq workflow | *Basics of Nextflow*

Passing parameters to nextflow script

- Any variable VAR defined as “params.VAR” can be set at execution using flags “--var”
- Its default value is set at declaration

Example script:

```
params.input_folder = "default_dir/"
Inputs = Channel.fromPath(params.input_folder)
...

> Nextflow run test.nf --input_folder BAM/
```

Part III. Coding an RNA-seq workflow | *Basics of Nextflow*

Configuring nextflow

- **nextflow.config files:** contains options
 - nextflow concatenates config file given using the “-c” flag with the *nextflow.config* from current directory, that from the workflow directory and the global config at \$HOME/.nextflow/config
 - useful to define *profiles* with options to run on different infrastructures
 - multiple profiles can be provided using the “-p” flag (e.g., “nextflow run -p conda -p SLURM_IARC”)

```
profiles {  
  conda {  
    process.conda = "$baseDir/environment.yml"  
  }  
  Singularity {  
    singularity.enabled = true  
    process.container =  
'shub://IARCBioinfo/fastqc-nf:v1.1'  
  }  
  SLURM_IARC {  
    process.executor = "slurm"  
    queue = "low_p"  
    queueSize = 50  
  }  
}
```

Part III. Coding an RNA-seq workflow | *Basics of Nextflow*

Configuring nextflow

- **nextflow.config files:** contains options for running nextflow.
- **Best practices for reproducibility:** create a conda recipe (yml file) containing all software dependencies

File environment.yml:

```
name: GATK4
channels:
  - bioconda
  - conda-forge
dependencies:
  - gatk4=4.1.5.0
```

Part III. Coding an RNA-seq workflow | *Basics of Nextflow*

Debugging a nextflow script

- **If error in the nextflow code:** usually an error is returned right away, processes are not executed; to debug: print values and channels (remember to duplicate with operator “.into()” so the printed channel is not consumed)
- **If error in script:** visit work directory created by nextflow and check files
 - `.command.sh`: actual script run
 - `.command.log`, `out`, `err`: log, output and error of script

```
name: N E X T F L O W ~ version 20.10.0
Launching `test.nf` [extravagant_heisenberg] - revision:
d3ac3534eb
executor > local (2)
[b9/d63363] process > test (2) [100%] 2 of 2, failed: 2✗
Error executing process > 'test (1)'
Caused by:
  Missing output file(s) `NA06984_T.bam.name` expected by
  process `test (1)`
Command executed:
  echo "{bam}" > {bam}.name
Command exit status:
  0
Command output:
  (empty)
Work dir:
/home/nalcala/Medical_Genomics_TP1/work/b1/9dc097b6e17e21852
file969075a92d
```


Part III. Coding an RNA-seq workflow | *Basics of Nextflow*

Resuming an execution

- The “-resume” flag restarts restarts the execution
- It checks which tasks were completed (they are cached)
- Processes with **changed parameters** will be re-run
- Processes with **modified code** will be re-run
- *Note: to be able to resume execution, nextflow keeps all intermediate files in its work directory, which needs to be cleaned periodically*

```
> Nextflow run test.nf -resume
```

Part III. Coding an RNA-seq workflow | *Write a workflow*

Practical 1. Questions 4-7

https://github.com/IARCbioinfo/medical_genomics_course/wiki/Practical-1

Part III. Coding an RNA-seq workflow | *Basics of Nextflow*

Being a nice user!

- **Computing facilities are shared resources, everyone needs to be mindful**

```
> export NXF_WORK=/temp/nalcala
```

Best practices:

- Work directory should be in the temporary folders (not \$HOME), e.g., scratch or temp folders
- Use nextflow environment variables to set the right paths
- Always specify “cpus” and “mem” directives, and adapt resource usage to the system

Part III. Coding an RNA-seq workflow | *Process the tumor data*

Practical 1. Questions 8

https://github.com/IARCbioinfo/medical_genomics_course/wiki/Practical-1

Part IV. Going further | *Advanced Nextflow*

Interaction with github

- github allows **open code sharing, versioning, automated tests**
- **Nextflow can use specific revisions from github, corresponding to branches, tags, or commit IDs**

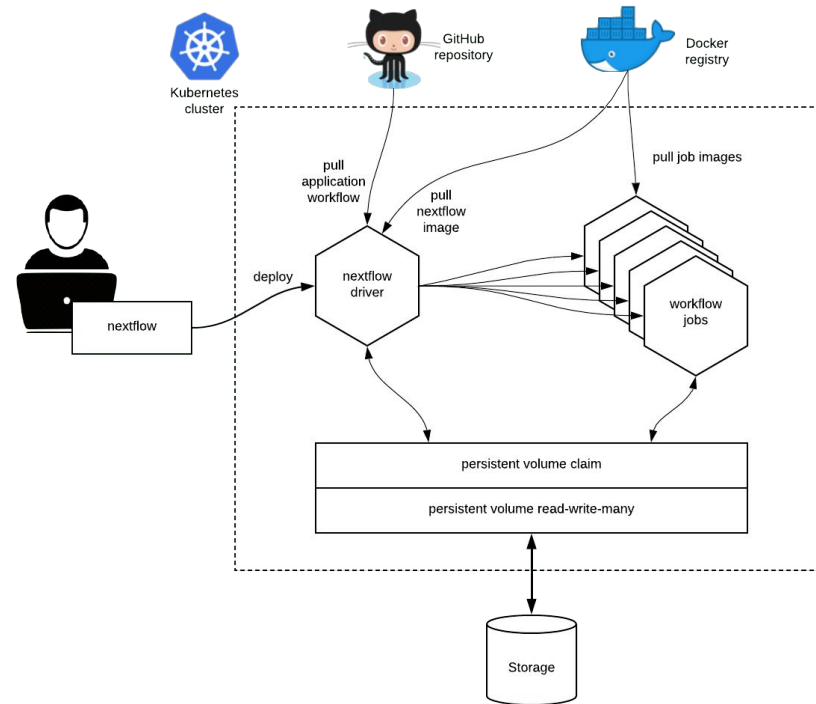
```
> Nextflow run IARCbioinfo/FastQC-nf -r v1.1  
--help
```

```
> Nextflow run IARCbioinfo/FastQC-nf -r  
06365b4dd152d586d2fb9e128c0188a5f84fd257 --help
```

Part IV. Going further | *Advanced Nextflow*

Use on other facilities

- Simply **setting the executor to the HPC scheduler** (e.g., Isf, slurm) enables using the HPC
- Can also be configured to use cloud resources (amazon web services, kubernetes, ...)



Part IV. Going further | *Advanced Nextflow*

An evolving language

- Nextflow is very recent, **syntax and features are changing rapidly**
- **DSL2** introduces **functional programming** allowing modules shared between workflows, reuse of channels, piping of processes, ...
- Nextflow tower.nf provides an interface to **monitor job execution** and now also **launch jobs** (similar to Galaxy)

```
nextflow.enable.dsl=2
```

```
include { foo } from './some/module1'  
include { bar } from './some/module2'
```

```
workflow {  
    channel.from('Hello') | map { it.reverse() } | (foo &  
bar) | mix | view  
}
```




Medical Genomics Practical #1:
Building reproducible workflows
International Agency for Research on Cancer
Lyon, France

Appendix

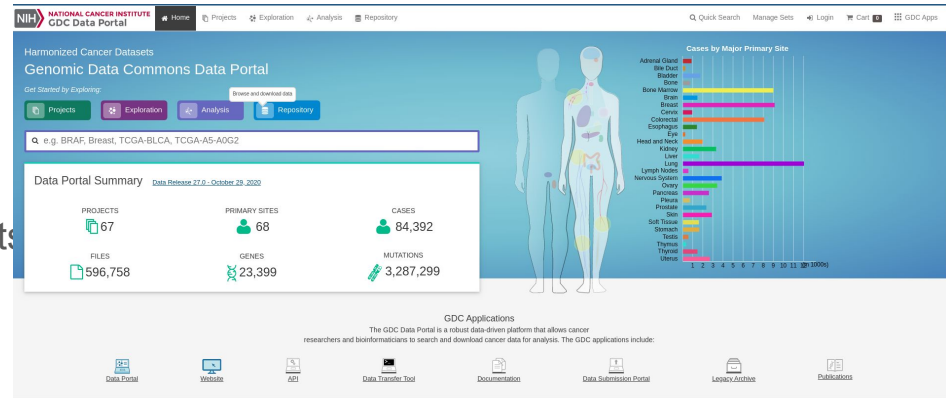
International Agency for Research on Cancer

Part I. Transcriptomics | Ressources

The Cancer Genome Atlas (TCGA) project

Database of cancer multi-omic data for

- Tumors from 33 primary sites
- RNA-seq data under controlled access (requires research institute affiliation)
- Processed gene expression data (read counts and FPKM) open-access



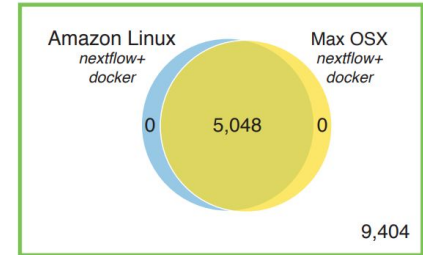
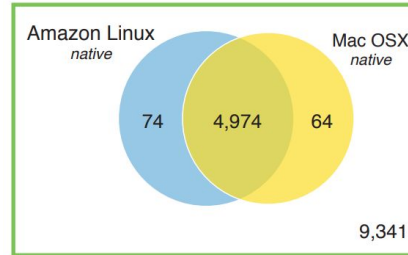
Web interface of the genomic data portal hosting the TCGA data. Source: <https://portal.gdc.cancer.gov/>.

Part I. Transcriptomics | *Techniques*

Bulk sequencing: processing

Notes:

- *RNA-seq analyses are known to suffer from a lack of robustness, so **reproducibility and open science practices are of the uttermost importance!***
- *Even subtle numerical instability issues can impact the results (**Figure**)*
- *See **Practical 1** for a solution to provide entirely reproducible workflows*



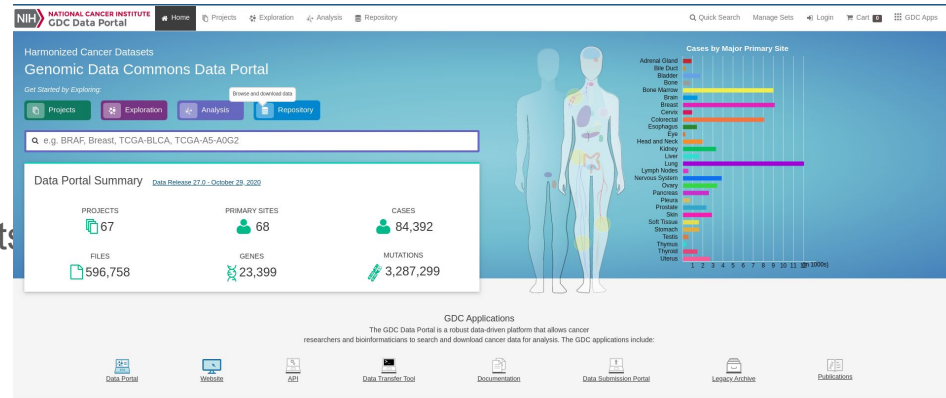
Reproducibility of RNA-seq differential expression analysis of Human Lung Fibroblasts. **Left.** Venn diagram of differentially expressed genes using the exact same versions of all softwares (kallisto and sleuth) but two different OS (blue: a Linux system from AWS, yellow: Mac OS). **Right.** Same as on the left, but running the entire data processing and analysis workflow inside a docker container using the Nextflow language. Source: Di Tommaso et al. *Nature Biotechnology* 2017.

Part I. Transcriptomics | Ressources

The Cancer Genome Atlas (TCGA) project

Database of cancer multi-omic data for

- Tumors from 33 primary sites
- RNA-seq data under controlled access (requires research institute affiliation)
- Processed gene expression data (read counts and FPKM) open-access



Web interface of the genomic data portal hosting the TCGA data. Source: <https://portal.gdc.cancer.gov/>.

Part I. Transcriptomics | Ressources

The Gene Expression Omnibus (GEO) repository

Database of expression data (arrays and RNA-seq)

- Includes human data
- All data is open-access

Will be used for the practicals.

The screenshot shows the NCBI Gene Expression Omnibus (GEO) website. At the top, there's a navigation bar with links for GEO Home, Documentation, Query & Browse, and Email GEO. A prominent red banner at the top right contains COVID-19 related information, including links to CDC and NIH resources. The main heading is 'Gene Expression Omnibus', followed by a brief description of the repository. Below this, there's a search bar with the placeholder text 'Keyword or GEO Accession' and a 'Search' button. The page is organized into three main columns: 'Getting Started' (with links like Overview, FAQ, About GEO DataSets, etc.), 'Tools' (with links like Search for Studies at GEO DataSets, Search for Gene Expression at GEO Profiles, etc.), and 'Browse Content' (with a table showing repository statistics). The 'Browse Content' table lists: DataSets: 4348, Series: 138500, Platforms: 21518, and Samples: 3983512. At the bottom, there's an 'Information for Submitters' section with links for Submission Guidelines, Update Guidelines, MIAME Standards, Citing and Linking to GEO, Guidelines for Reviewers, and GEO Publications.

Browse Content	
Repository Browser	
DataSets:	4348
Series:	138500
Platforms:	21518
Samples:	3983512

Web interface of the gene expression omnibus repository. Source: <https://www.ncbi.nlm.nih.gov/geo/>.