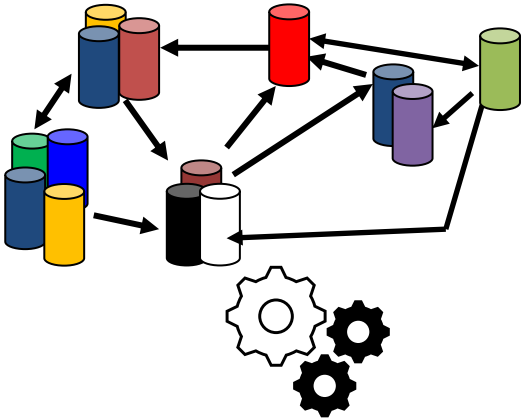
10 ans après la crise de la reproductibilité en bioinformatique : de la reproduction à la réutilisationde workflows scientifiques

Sarah Cohen-Boulakia November, 2025

Laboratoire Interdisciplinaire des Sciences du Numérique – LISN
Université Paris-Saclay

Biological and Biomedical data analysis



Public sources

Distributed

Heterogeneous Network

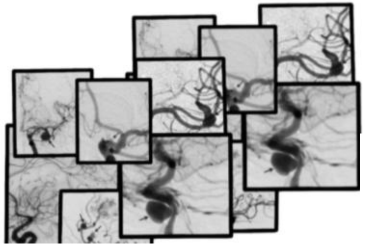
> 1,500 (NAR)

Tools – Scripts

Distributed

Heterogeneous

> 23,000 (bio.tools)



Which exact dataset did I use? Which tool version? Which parameters…?

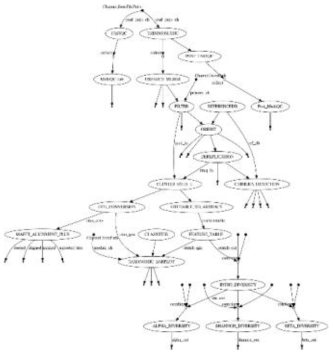
Analysis pipelines

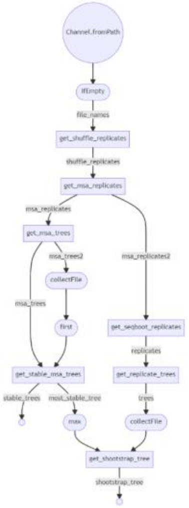
Combining multiple tools

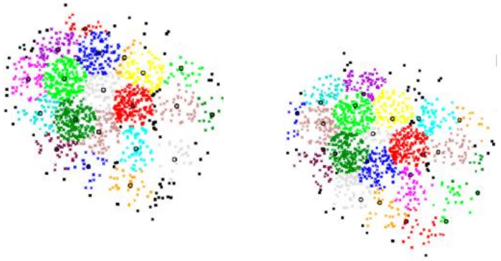
Heterogeneous

Various environnements & platforms

> 1,000 workflows (GitHub)







Studies on reproducibility - Reproducibility Crisis...

Nekrutenko & Taylor - Nature Genetics (2012)

50 papers 2011 using the Burrows-Wheeler Aligner

31/50 (62%) provide no information

no version of the tool + no parameters + no genomic ref sequence

7/50 (14%) provide all the necessary details

Alsheikh-Ali et al, PLoS one (2011)

10 papers in the top-50 IF journals → 500 papers

149 (30%) were not subject to any data availability policy

(0% data available)

Of the remaining 351 papers

208 papers (59%) did not adhere to the data availability instructions

143 make a statement of *willingness* to share

47 papers (9%) deposited full primary raw data online

→ Computational reproducibility

Studies on reproducibility - Reproducibility Crisis…

Nekrutenko & Taylor - Nature Genetics (2012)

50 papers 2011 using the Burrows-Wheeler Aligner

31/50 (62%) provide no information

no version of the tool + no parameters + no genomic ref sequence

7/50 (14%) provide all the necessary details

Alsheikh-Ali et al, PLoS one (2011)

10 papers in the top-50 IF journals → 500 papers

149 (30%) were not subject to any data availability policy

(0% data available)

Of the remaining 351 papers

208 papers (59%) did not adhere to the data availability instructions

143 make a statement of *willingness* to share

47 papers (9%) deposited full primary raw data online

→ Computational reproducibility

Raise standards for preclinical cancer research

47/53 "landmark" publications could not be replicated

[Begley, Ellis Nature, 483, 2012]

Must try harder

Error prone

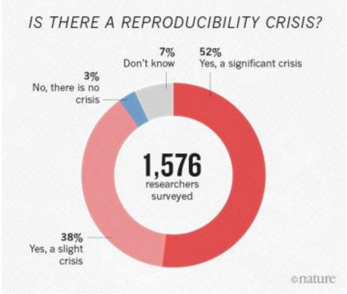
Biologists must realize the pitfalls … massive amounts of data.

If a job is worth doing, it is worth doing twice

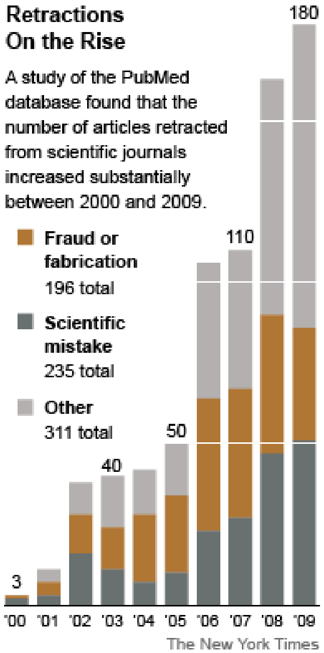
The case for open computer programs

Six red flags for suspect work

Know when your numbers are significant



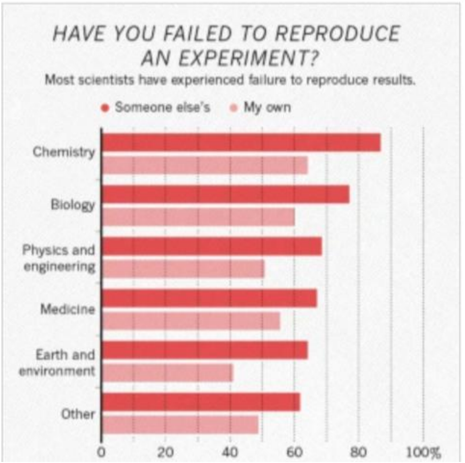
IS THERE A REPRODUCIBILITY CRISIS?



Retractions On the Rise

A study of the PubMed database found that the number of articles retracted from scientific journals increased substantially between 2000 and 2009.

The New York Times



HAVE YOU FAILED TO REPRODUCE AN EXPERIMENT?

Most scientists have experienced failure to reproduce results.

Current solutions to computational reproductibility

Scientific Workflow Management Systems

“Data analysis pipeline ”

Data flow driven

Galaxy

snakemake

nextflow

WF specification

connected *processors*

steps of the analysis

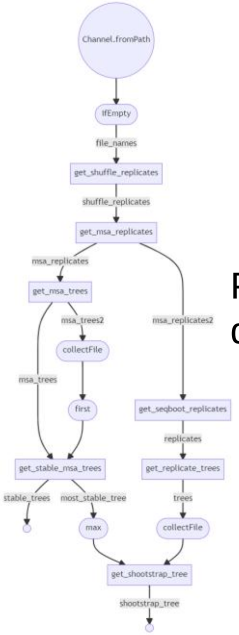
WF execution

data consumed/produced

Provenance modules

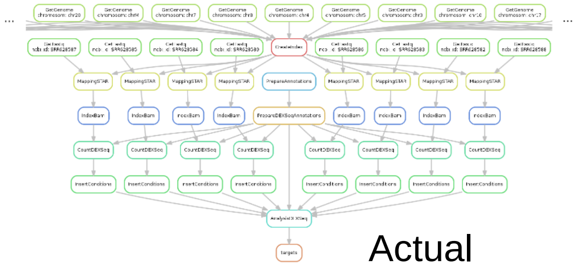
Scheduling …

Specification Steps



Possible dataflow

Executed Steps


...
GetGenome chromosom: chr20
GetGenome chromosom: chrM
GetGenome chromosom: chr7
GetGenome chromosom: chr8
GetGenome chromosom: chr4
GetGenome chromosom: chr5
GetGenome chromosom: chr3
GetGenome chromosom: chr10
GetGenome chromosom: chr17
...
GetFastq ncbi_id: SRR628587
GetFastq ncbi_id: SRR628585
GetFastq ncbi_id: SRR628584
GetFastq ncbi_id: SRR628589
CreateIndex
GetFastq ncbi_id: SRR628586
GetFastq ncbi_id: SRR628583
GetFastq ncbi_id: SRR628582
GetFastq ncbi_id: SRR628588
MappingSTAR
PrepareAnnotations
IndexBam
PrepareDEXSeqAnnotations
CountDEXSeq
InsertConditions
AnalysisDEXSeq
targets

Actual dataflow

Current solutions to computational reproductibility

Scientific Workflow Management Systems

“Data analysis pipeline ”

Data flow driven

WF specification

connected *processors*

steps of the analysis

WF execution

data consumed/produced

Provenance modules

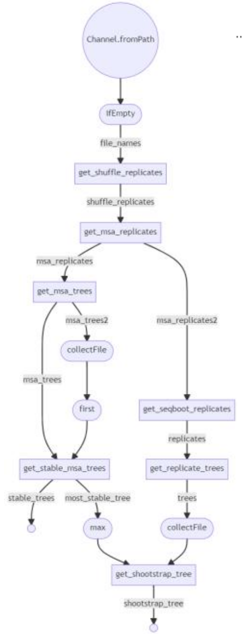
Scheduling …

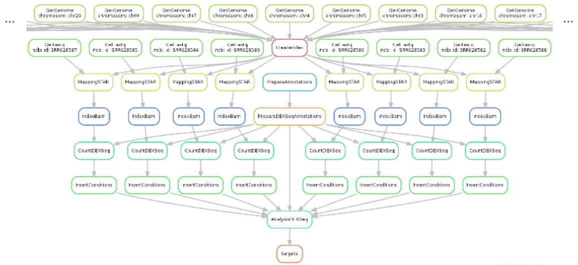
WF environment

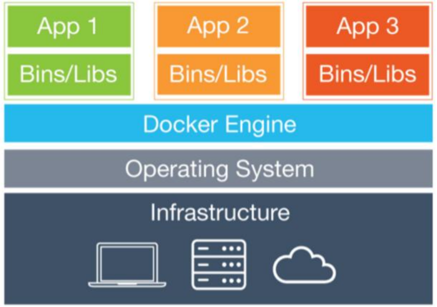
everything needed to run Libraries, dependencies…

Coupling WF with Docker/AppTainer

BioContainer




...
GetGenome chromosom: chr20
GetGenome chromosom: chrM
GetGenome chromosom: chr7
GetGenome chromosom: chr8
GetGenome chromosom: chr4
GetGenome chromosom: chr5
GetGenome chromosom: chr3
GetGenome chromosom: chr10
GetGenome chromosom: chr17
...
GetFastq ncbi_id: SRR628587
GetFastq ncbi_id: SRR628585
GetFastq ncbi_id: SRR628584
GetFastq ncbi_id: SRR628589
CreateIndex
GetFastq ncbi_id: SRR628586
GetFastq ncbi_id: SRR628583
GetFastq ncbi_id: SRR628582
GetFastq ncbi_id: SRR628588
MappingSTAR
PrepareAnnotations
IndexBam
PrepareDEXSeqAnnotations
CountDEXSeq
InsertConditions
AnalysisDEXSeq
targets


App 1
App 2
App 3
Bins/Libs
Bins/Libs
Bins/Libs
Docker Engine
Operating System
Infrastructure

Levels of computational reproducibility

From identical output to the same scientific results

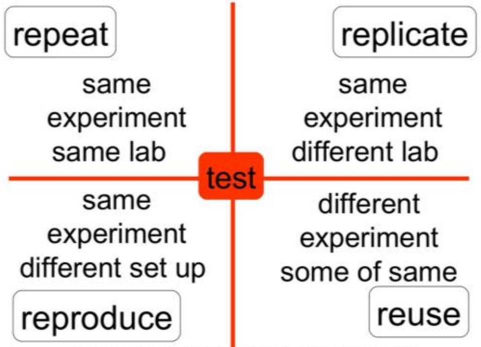
Repeat

Redo - exact same context

Same workflow, execution setting, environment

Identical *output*

Aim = proof for reviewers ☺



repeat

same experiment same lab

replicate

same experiment different lab

test

reproduce

same experiment different set up

reuse

different experiment some of same

Drummond C Replicability is not Reproducibility: Nor is it Good Science, online
Peng RD, Reproducible Research in Computational Science *Science 2 Dec 2011: 1226-1227.*

Replicate

Variation allowed in the workflows execution setting environment

Similar *output*

Aim = robustness

Levels of computational reproducibility

From identical output to the same scientific results

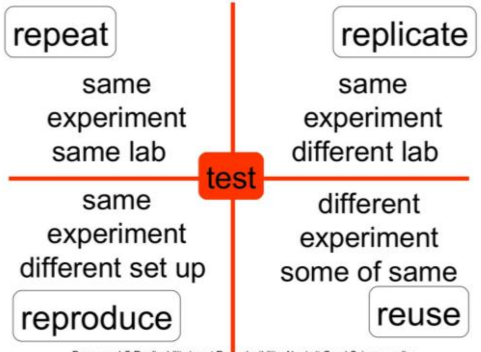
Repeat

Redo - exact same context

Same workflow, execution setting, environment

Identical *output*

Aim = proof for reviewers ☺



Drummond C Replicability is not Reproducibility: Nor is it Good Science, online
Peng RD, Reproducible Research in Computational Science *Science 2 Dec 2011: 1226-1227.*

Replicate

Variation allowed in the workflows execution setting environment

Similar *output*

Aim = robustness

Reproduce

Same *scientific result*

But the means used may be changed

Different workflows, execution setting, environment

Different output but in accordance with the result

Reuse

Adapt to new needs

Possibly different scientific result

Reuse in part existing workflows

→ Cumulative science

→ No Reuse without Repeat!

Reproducibility Networks

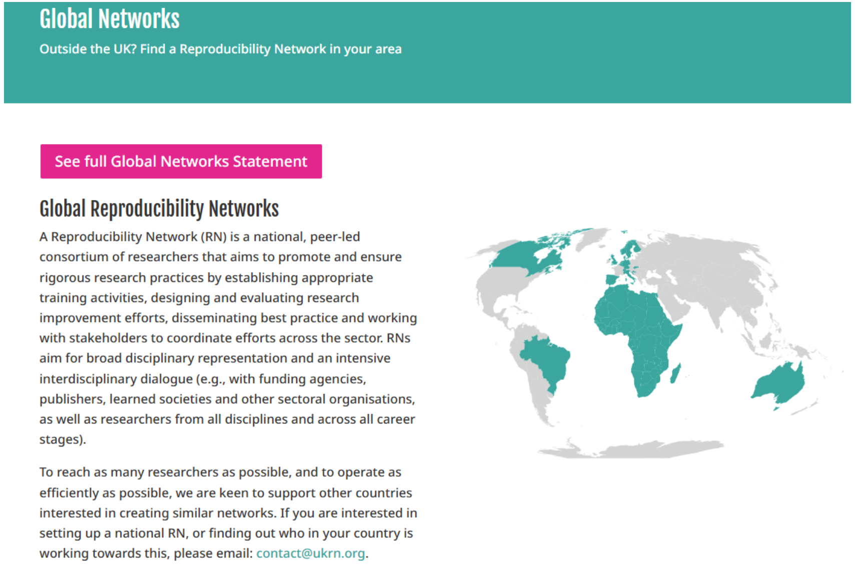
FR Reproducible Research network

French Reproducibility Network born in 2023
270+ colleagues

Next Scientific Days in Lyon
April 3-4th

https://www.recherche-reproductible.fr/index-en

MINISTÈRE
DE L'ENSEIGNEMENT
SUPÉRIEUR
ET DE LA RECHERCHE
Liberté
Égalité
Fraternité



Global Networks

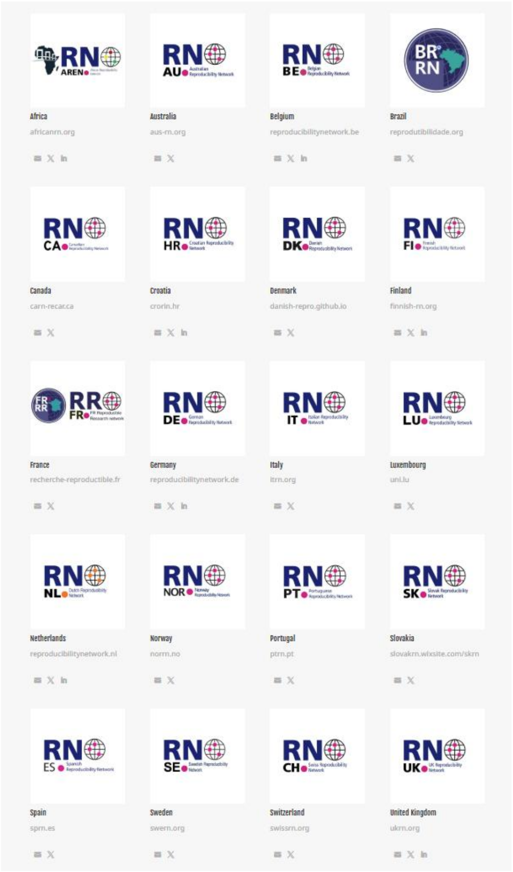
Outside the UK? Find a Reproducibility Network in your area

See full Global Networks Statement

Global Reproducibility Networks

A Reproducibility Network (RN) is a national, peer-led consortium of researchers that aims to promote and ensure rigorous research practices by establishing appropriate training activities, designing and evaluating research improvement efforts, disseminating best practice and working with stakeholders to coordinate efforts across the sector. RNs aim for broad disciplinary representation and an intensive interdisciplinary dialogue (e.g., with funding agencies, publishers, learned societies and other sectoral organisations, as well as researchers from all disciplines and across all career stages).

To reach as many researchers as possible, and to operate as efficiently as possible, we are keen to support other countries interested in creating similar networks. If you are interested in setting up a national RN, or finding out who in your country is working towards this, please email: contact@ukrn.org.



Africa	africanrn.org
Australia	aus-rn.org
Belgium	reproducibilitynetwork.be
Brazil	reprodutibilidade.org
Canada	carn-recar.ca
Croatia	crorin.hr
Denmark	danish-repro.github.io
Finland	finnish-rn.org
France	recherche-reproductible.fr
Germany	reproducibilitynetwork.de
Italy	itrn.org
Luxembourg	uni.lu
Netherlands	reproducibilitynetwork.nl
Norway	norrn.no
Portugal	ptrn.pt
Slovakia	slovakrn.wixsite.com/skrn
Spain	sprn.es
Sweden	swern.org
Switzerland	swissrn.org
United Kingdom	ukrn.org

Outline

The reproducibility landscape

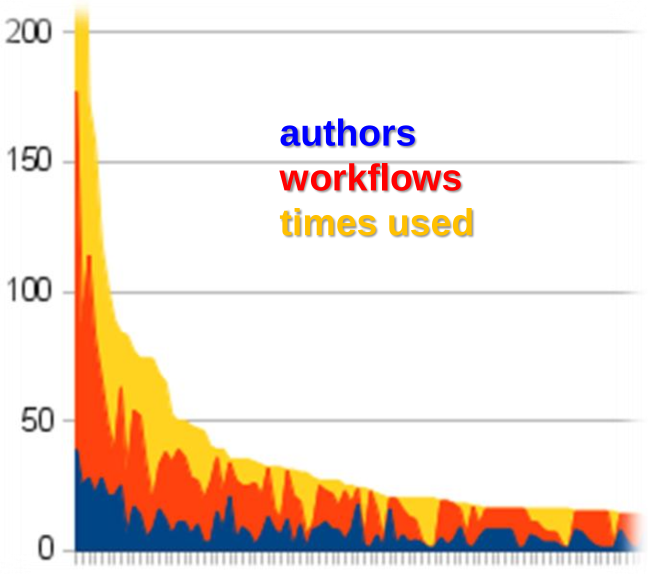
Status of workflow reuse

How to improve workflow reuse?

ZOOM on 2 current contributions

Conclusion

Reuse status - 2010



Distinct processors

1,700 Taverna workflows
10 242 processors (analysis steps)
Centralized in myExperiment

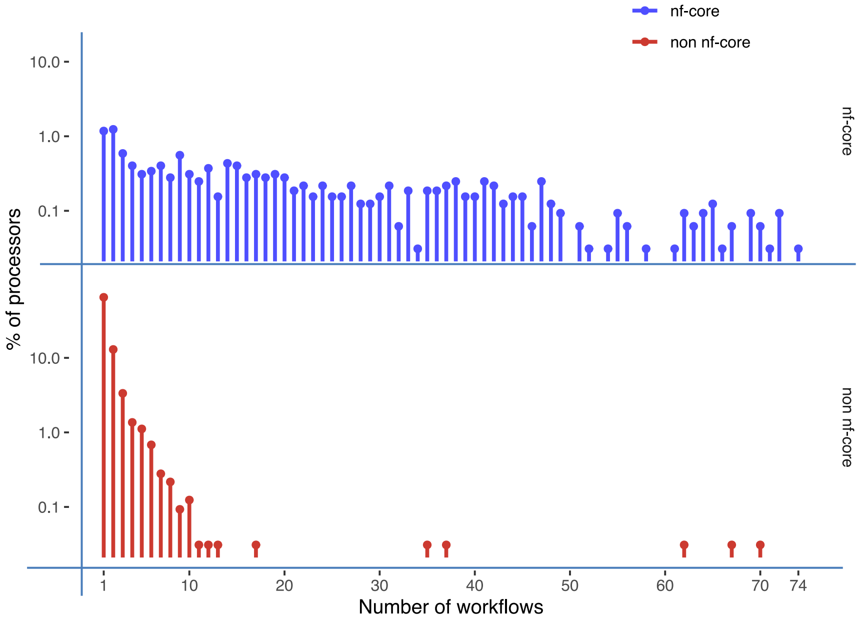
Re-use rates have a Zipf-like distribution

The top ten authors published 62% of all workflows

Reuse status - 2023

2,443 workflows Nextflow & Snakemake
15 540 processors (analysis steps)
Distributed in github
nf-core: repro of carefully checked Nextflow processors

Marine **Djaffardjy**



Still low reuse

The top ten authors published only 15% of all workflows

Higher reuse (black box) of processors in nf-core

université PARIS-SACLAY
INSTITUT DATAIA Science des données, Intelligence & Société

Outline

The reproducibility landscape

Status of workflow reuse

How to improve workflow reuse?

ZOOM on 2 current contributions

Conclusion

ShareFAIR

ShareFAIR : Sharing reliable workflows to transform datasets into gold standards: Application to Neuro-Vascular Pathologies

PEPR Santé Numérique

Coordinator

- Sarah Cohen-Boulakia

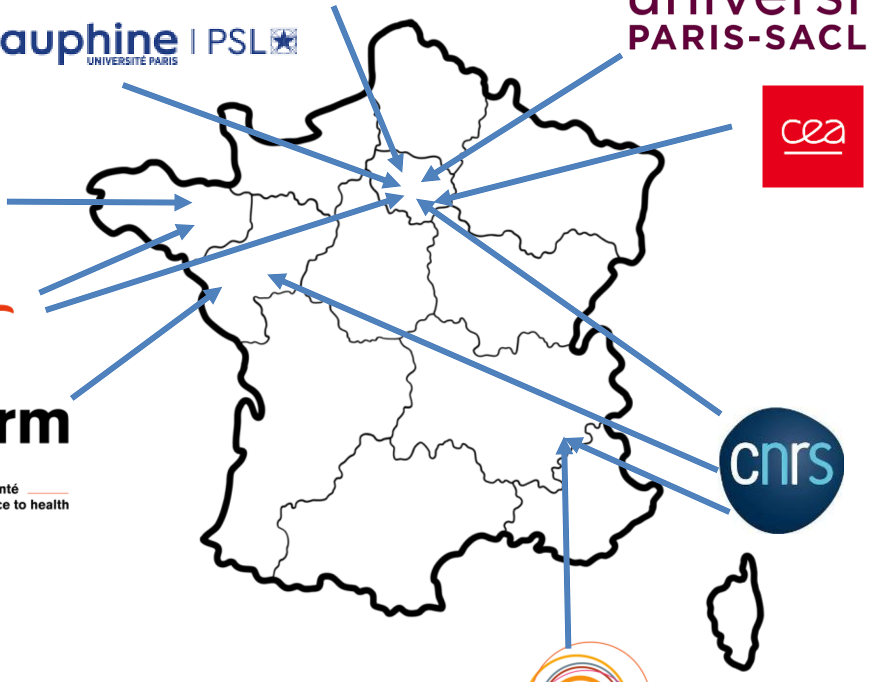
Managment structure

- Université Paris-Saclay

Partners

- Université Paris Dauphine PSL
- Institut Pasteur
- Université Lyon
- Université Rennes
- Inria
- CEA
- CNRS, INSERM (ITX)

Duration: 48 mois

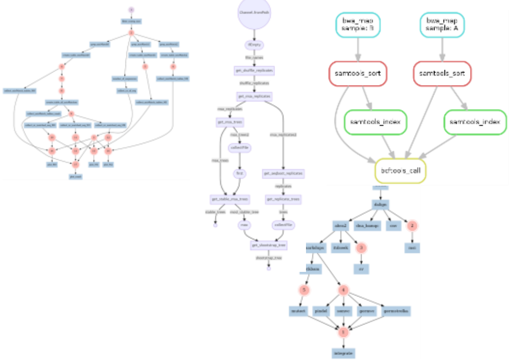


FRANCE 2030

université PARIS-SACLAY | INSTITUT DATAIA Science des données, Intelligence & Société

Objectives (1/3)

Define an interoperable framework for the design, annotation, and sharing of reproducible and reuseable workflows



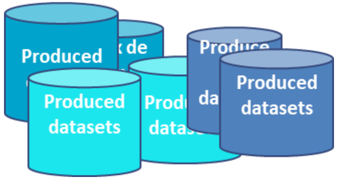
Design a **language** to query workflows, and their executions

Capture provenance (executions) in an optimized manner

Develop a **FAIRification method** for datasets

Guide workflow developers in **discovering and comparing** workflows

Execution traces: provenance of produced data



Objectives (2/3)

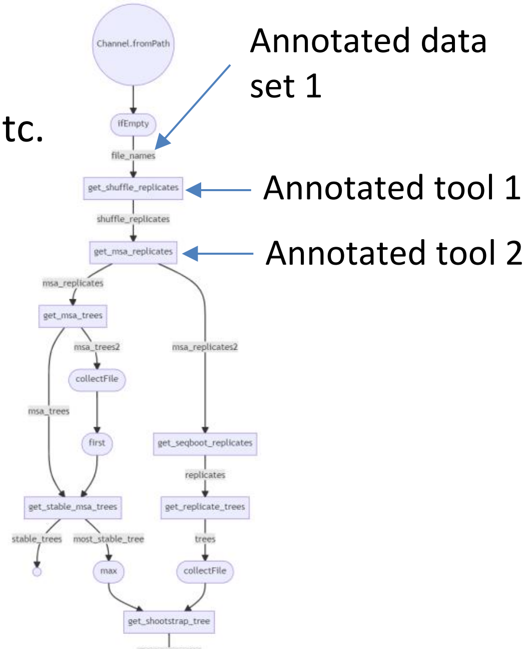
***Complete standards* for uniformly annotating workflows in terms of analysis tools and input/output datasets to produce FAIR datasets**

Identification of a set of existing standards for annotating protocols, workflows, and datasets: EDAM, EDAM-BioImaging, MONDO, DUO, etc.

Identification of overlaps, varying levels of precision, and missing concepts

Construction of annotation guides

Development of a Knowledge base



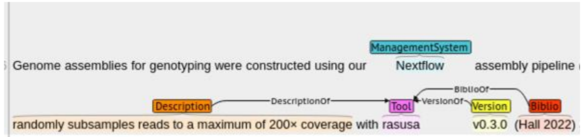
Annotated data set 1

Annotated tool 1

Annotated tool 2

Objectives (3/3)

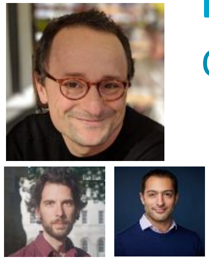
***Augment the set of workflows* by extracting them from text (litterature) and large datasets from communauties**



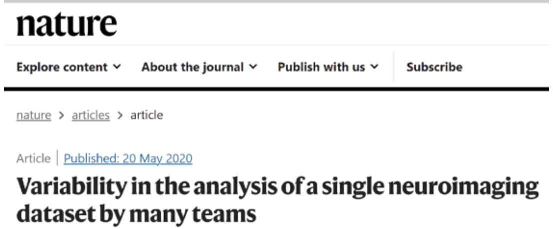
Genome assemblies for genotyping were constructed using our Nextflow assembly pipeline

randomly subsamples reads to a maximum of 200× coverage with rasusa v0.3.0 (Hall 2022)

Develop NLP models to extract workflow description from the scientific literature



Learn protocols form clinical data collected all along the care activity (patient data). Comparision between learnt and declared protocols.



nature

Article | Published: 20 May 2020

Variability in the analysis of a single neuroimaging dataset by many teams

Extract protocols and workflows from large shared datasets in neuroimagery

Outline

The reproducibility landscape

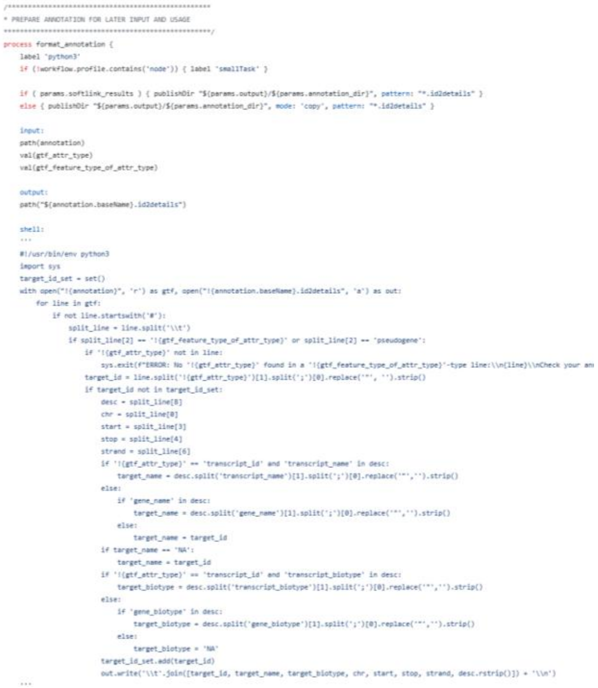
Status of workflow reuse

How to improve workflow reuse

ZOOM on 2 current contributions

Conclusion

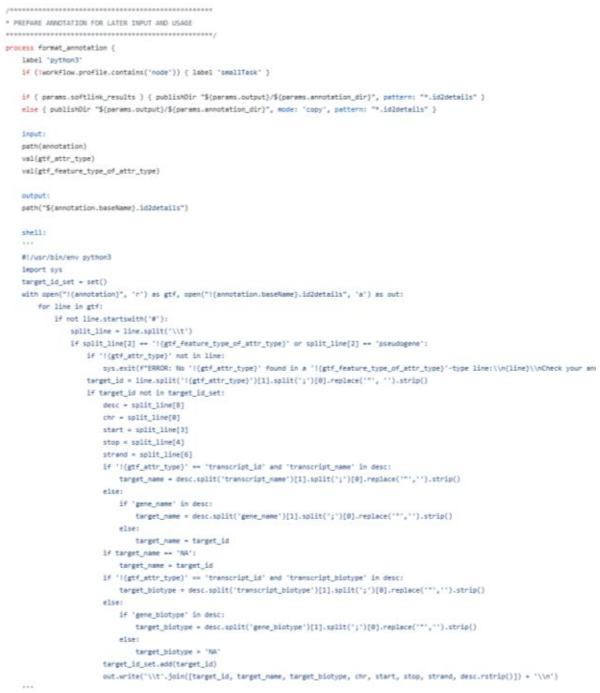
Exploiting the 3 forms of the workflow



```
/****************************************************
* PREPARE ANNOTATION FOR LATER INPUT AND USAGE
****************************************************/
process format_annotation {
    label 'python3'
    if (!workflow.profile.contains('node')) { label 'smallTask' }

    if ( params.softlink_results ) { publishDir "${params.output}/${params.annotation_dir}", pattern: "*.id2details" }
    else { publishDir "${params.output}/${params.annotation_dir}", mode: 'copy', pattern: "*.id2details" }

    input:
    path(annotation)
    val(gtf_attr_type)
    val(gtf_feature_type_of_attr_type)

    output:
    path("${annotation.baseName}.id2details")

    shell:
    '''
    #!/usr/bin/env python3
    import sys
    target_id_set = set()
    with open("!{annotation}", 'r') as gtf, open("!{annotation.baseName}.id2details", 'a') as out:
        for line in gtf:
            if not line.startswith('#'):
                split_line = line.split('\\t')
                if split_line[2] == '!{gtf_feature_type_of_attr_type}' or split_line[2] == 'pseudogene':
                    if '!{gtf_attr_type}' not in line:
                        sys.exit(f"ERROR: No '!{gtf_attr_type}' found in a '!{gtf_feature_type_of_attr_type}'-type line:\\n{line}\\nCheck your an
                    target_id = line.split('!{gtf_attr_type}')[1].split(';')[0].replace('"', '').strip()
                    if target_id not in target_id_set:
                        desc = split_line[8]
                        chr = split_line[0]
                        start = split_line[3]
                        stop = split_line[4]
                        strand = split_line[6]
                        if '!{gtf_attr_type}' == 'transcript_id' and 'transcript_name' in desc:
                            target_name = desc.split('transcript_name')[1].split(';')[0].replace('"','').strip()
                        else:
                            if 'gene_name' in desc:
                                target_name = desc.split('gene_name')[1].split(';')[0].replace('"','').strip()
                            else:
                                target_name = target_id
                        if target_name == 'NA':
                            target_name = target_id
                        if '!{gtf_attr_type}' == 'transcript_id' and 'transcript_biotype' in desc:
                            target_biotype = desc.split('transcript_biotype')[1].split(';')[0].replace('"','').strip()
                        else:
                            if 'gene_biotype' in desc:
                                target_biotype = desc.split('gene_biotype')[1].split(';')[0].replace('"','').strip()
                            else:
                                target_biotype = 'NA'
                        target_id_set.add(target_id)
                        out.write('\\t'.join([target_id, target_name, target_biotype, chr, start, stop, strand, desc.rstrip()]) + '\\n')
    '''
```

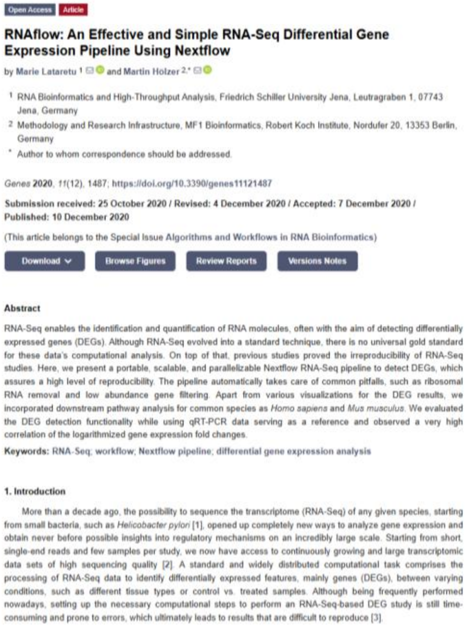
Code: GitHub - data science

Exploiting the 3 forms of the workflow



```
/*****************************************************
* PREPARE ANNOTATION FOR LATER INPUT AND USAGE
*****************************************************/
process format_annotation {
    label 'python3'
    if (!workflow.profile.contains('node')) { label 'smallTask' }

    if ( params.softlink_results ) { publishDir "${params.output}/${params.annotation_dir}", pattern: "*.id2details" }
    else { publishDir "${params.output}/${params.annotation_dir}", mode: 'copy', pattern: "*.id2details" }

    input:
    path(annotation)
    val(gtf_attr_type)
    val(gtf_feature_type_of_attr_type)

    output:
    path("${annotation.baseName}.id2details")

    shell:
    '''
    #!/usr/bin/env python3
    import sys
    target_id_set = set()
    with open("!{annotation}", 'r') as gtf, open("!{annotation.baseName}.id2details", 'a') as out:
        for line in gtf:
            if not line.startswith('#'):
                split_line = line.split('\\t')
                if split_line[2] == '!{gtf_feature_type_of_attr_type}' or split_line[2] == 'pseudogene':
                    if '!{gtf_attr_type}' not in line:
                        sys.exit(f"ERROR: No '!{gtf_attr_type}' found in a '!{gtf_feature_type_of_attr_type}'-type line:\\n{line}\\nCheck your an
                    target_id = line.split('!{gtf_attr_type}')[1].split(';')[0].replace('"', '').strip()
                    if target_id not in target_id_set:
                        desc = split_line[8]
                        chr = split_line[0]
                        start = split_line[3]
                        stop = split_line[4]
                        strand = split_line[6]
                        if '!{gtf_attr_type}' == 'transcript_id' and 'transcript_name' in desc:
                            target_name = desc.split('transcript_name')[1].split(';')[0].replace('"','').strip()
                        else:
                            if 'gene_name' in desc:
                                target_name = desc.split('gene_name')[1].split(';')[0].replace('"','').strip()
                            else:
                                target_name = target_id
                        if target_name == 'NA':
                            target_name = target_id
                        if '!{gtf_attr_type}' == 'transcript_id' and 'transcript_biotype' in desc:
                            target_biotype = desc.split('transcript_biotype')[1].split(';')[0].replace('"','').strip()
                        else:
                            if 'gene_biotype' in desc:
                                target_biotype = desc.split('gene_biotype')[1].split(';')[0].replace('"','').strip()
                            else:
                                target_biotype = 'NA'
                        target_id_set.add(target_id)
                        out.write('\\t'.join([target_id, target_name, target_biotype, chr, start, stop, strand, desc.rstrip()]) + '\\n')
    '''
```

Code: GitHub - data science



RNAflow: An Effective and Simple RNA-Seq Differential Gene Expression Pipeline Using Nextflow

by Marie Lataretu [1] and Martin Holzer [2,*]

[1] RNA Bioinformatics and High-Throughput Analysis, Friedrich Schiller University Jena, Leutragraben 1, 07743 Jena, Germany

[2] Methodology and Research Infrastructure, MF1 Bioinformatics, Robert Koch Institute, Nordufer 20, 13353 Berlin, Germany

[*] Author to whom correspondence should be addressed.

Genes **2020**, *11*(12), 1487; https://doi.org/10.3390/genes11121487

Submission received: 25 October 2020 / Revised: 4 December 2020 / Accepted: 7 December 2020 / Published: 10 December 2020

(This article belongs to the Special Issue Algorithms and Workflows in RNA Bioinformatics)

Abstract

RNA-Seq enables the identification and quantification of RNA molecules, often with the aim of detecting differentially expressed genes (DEGs). Although RNA-Seq evolved into a standard technique, there is no universal gold standard for these data's computational analysis. On top of that, previous studies proved the irreproducibility of RNA-Seq studies. Here, we present a portable, scalable, and parallelizable Nextflow RNA-Seq pipeline to detect DEGs, which assures a high level of reproducibility. The pipeline automatically takes care of common pitfalls, such as ribosomal RNA removal and low abundance gene filtering. Apart from various visualizations for the DEG results, we incorporated downstream pathway analysis for common species as *Homo sapiens* and *Mus musculus*. We evaluated the DEG detection functionality while using qRT-PCR data serving as a reference and observed a very high correlation of the logarithmized gene expression fold changes.

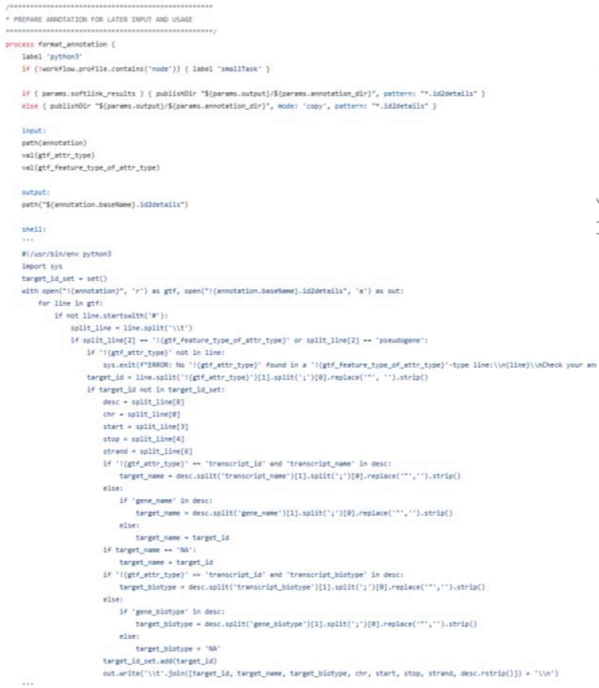
Keywords: RNA-Seq; workflow; Nextflow pipeline; differential gene expression analysis

1. Introduction

More than a decade ago, the possibility to sequence the transcriptome (RNA-Seq) of any given species, starting from small bacteria, such as *Helicobacter pylori* [1], opened up completely new ways to analyze gene expression and obtain never before possible insights into regulatory mechanisms on an incredibly large scale. Starting from short, single-end reads and few samples per study, we now have access to continuously growing and large transcriptomic data sets of high sequencing quality [2]. A standard and widely distributed computational task comprises the processing of RNA-Seq data to identify differentially expressed features, mainly genes (DEGs), between varying conditions, such as different tissue types or control vs. treated samples. Although being frequently performed nowadays, setting up the necessary computational steps to perform an RNA-Seq-based DEG study is still time-consuming and prone to errors, which ultimately leads to results that are difficult to reproduce [3].

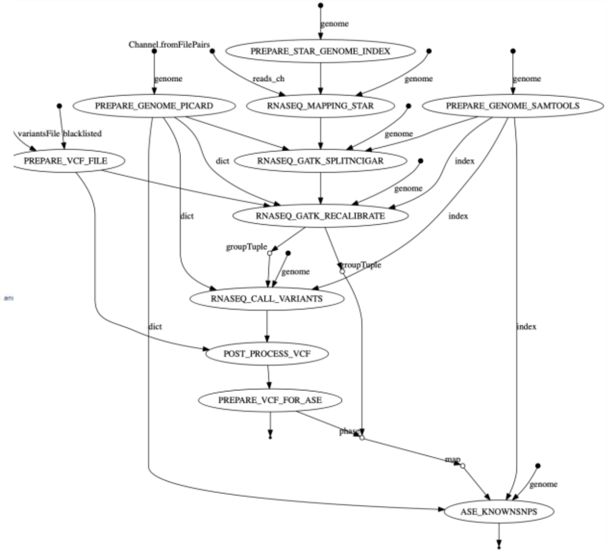
Text: scientific papers

Exploiting the 3 forms of the workflow

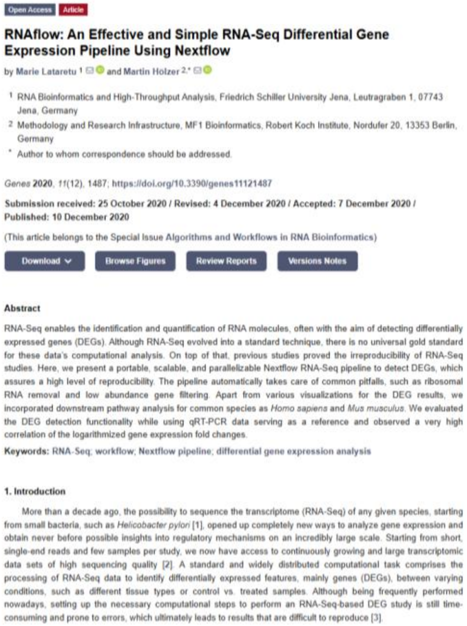


```
/**************************************************
* PREPARE ANNOTATION FOR LATER INPUT AND USAGE
**************************************************/
process format_annotation {
    label 'python3'
    if (!workflow.profile.contains('node')) { label 'smallTask' }

    if ( params.softlink_results ) { publishDir "${params.output}/${params.annotation_dir}", pattern: "*.id2details" }
    else { publishDir "${params.output}/${params.annotation_dir}", mode: 'copy', pattern: "*.id2details" }

    input:
    path(annotation)
    val(gtf_attr_type)
    val(gtf_feature_type_of_attr_type)

    output:
    path("${annotation.baseName}.id2details")

    shell:
    '''
    #!/usr/bin/env python3
    import sys
    target_id_set = set()
    with open("!{annotation}", 'r') as gtf, open("!{annotation.baseName}.id2details", 'a') as out:
        for line in gtf:
            if not line.startswith('#'):
                split_line = line.split('\\t')
                if split_line[2] == '!{gtf_feature_type_of_attr_type}' or split_line[2] == 'pseudogene':
                    if '!{gtf_attr_type}' not in line:
                        sys.exit(f"ERROR: No '!{gtf_attr_type}' found in a '!{gtf_feature_type_of_attr_type}'-type line:\\n{line}\\nCheck your an
                    target_id = line.split('!{gtf_attr_type}')[1].split(';')[0].replace('"', '').strip()
                    if target_id not in target_id_set:
                        desc = split_line[8]
                        chr = split_line[0]
                        start = split_line[3]
                        stop = split_line[4]
                        strand = split_line[6]
                        if '!{gtf_attr_type}' == 'transcript_id' and 'transcript_name' in desc:
                            target_name = desc.split('transcript_name')[1].split(';')[0].replace('"','').strip()
                        else:
                            if 'gene_name' in desc:
                                target_name = desc.split('gene_name')[1].split(';')[0].replace('"','').strip()
                            else:
                                target_name = target_id
                        if target_name == 'NA':
                            target_name = target_id
                        if '!{gtf_attr_type}' == 'transcript_id' and 'transcript_biotype' in desc:
                            target_biotype = desc.split('transcript_biotype')[1].split(';')[0].replace('"','').strip()
                        else:
                            if 'gene_biotype' in desc:
                                target_biotype = desc.split('gene_biotype')[1].split(';')[0].replace('"','').strip()
                            else:
                                target_biotype = 'NA'
                        target_id_set.add(target_id)
                        out.write('\\t'.join([target_id, target_name, target_biotype, chr, start, stop, strand, desc.rstrip()]) + '\\n')
    '''
```

Code: GitHub - data science



Graph: workflow structure



Open Access Article

RNAflow: An Effective and Simple RNA-Seq Differential Gene Expression Pipeline Using Nextflow

by Marie Lataretu [1] and Martin Hölzer [2,*]

[1] RNA Bioinformatics and High-Throughput Analysis, Friedrich Schiller University Jena, Leutragraben 1, 07743 Jena, Germany
[2] Methodology and Research Infrastructure, MF1 Bioinformatics, Robert Koch Institute, Nordufer 20, 13353 Berlin, Germany
[*] Author to whom correspondence should be addressed.

Genes **2020**, *11*(12), 1487; https://doi.org/10.3390/genes11121487

Submission received: 25 October 2020 / Revised: 4 December 2020 / Accepted: 7 December 2020 / Published: 10 December 2020

(This article belongs to the Special Issue Algorithms and Workflows in RNA Bioinformatics)

Abstract

RNA-Seq enables the identification and quantification of RNA molecules, often with the aim of detecting differentially expressed genes (DEGs). Although RNA-Seq evolved into a standard technique, there is no universal gold standard for these data's computational analysis. On top of that, previous studies proved the irreproducibility of RNA-Seq studies. Here, we present a portable, scalable, and parallelizable Nextflow RNA-Seq pipeline to detect DEGs, which assures a high level of reproducibility. The pipeline automatically takes care of common pitfalls, such as ribosomal RNA removal and low abundance gene filtering. Apart from various visualizations for the DEG results, we incorporated downstream pathway analysis for common species as *Homo sapiens* and *Mus musculus*. We evaluated the DEG detection functionality while using qRT-PCR data serving as a reference and observed a very high correlation of the logarithmized gene expression fold changes.

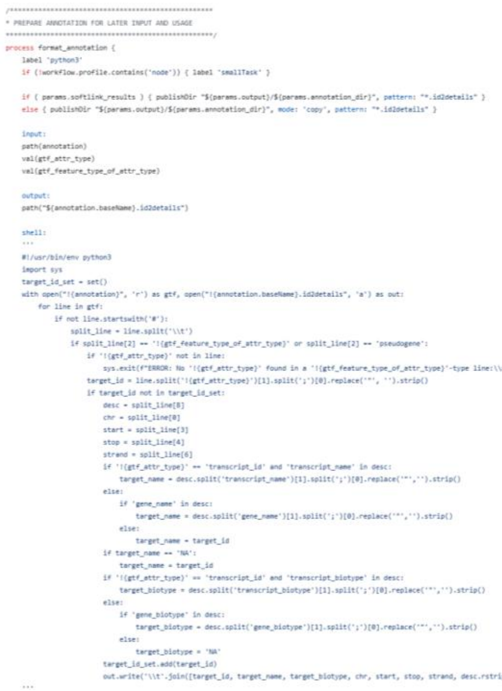
Keywords: RNA-Seq; workflow; Nextflow pipeline; differential gene expression analysis

1. Introduction

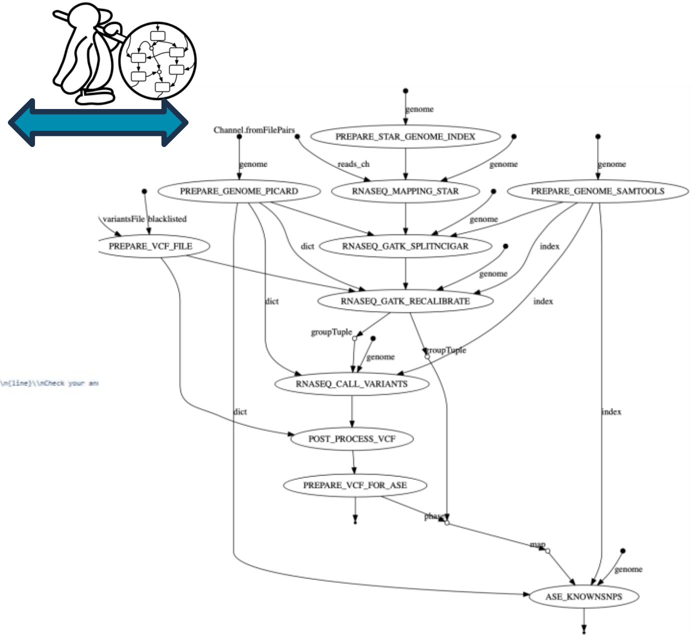
More than a decade ago, the possibility to sequence the transcriptome (RNA-Seq) of any given species, starting from small bacteria, such as *Helicobacter pylori* [1], opened up completely new ways to analyze gene expression and obtain never before possible insights into regulatory mechanisms on an incredibly large scale. Starting from short, single-end reads and few samples per study, we now have access to continuously growing and large transcriptomic data sets of high sequencing quality [2]. A standard and widely distributed computational task comprises the processing of RNA-Seq data to identify differentially expressed features, mainly genes (DEGs), between varying conditions, such as different tissue types or control vs. treated samples. Although being frequently performed nowadays, setting up the necessary computational steps to perform an RNA-Seq-based DEG study is still time-consuming and prone to errors, which ultimately leads to results that are difficult to reproduce [3].

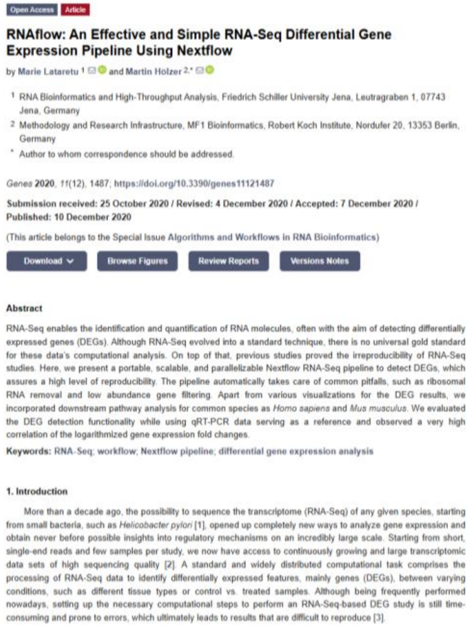
Text: scientific papers

Exploiting the 3 forms of the workflow



```
/***************************************************
* PREPARE ANNOTATION FOR LATER INPUT AND USAGE
***************************************************/
process format_annotation {
    label 'python3'
    if (!workflow.profile.contains('node')) { label 'smallTask' }

    if ( params.softlink_results ) { publishDir "${params.output}/${params.annotation_dir}", pattern: "*.id2details" }
    else { publishDir "${params.output}/${params.annotation_dir}", mode: 'copy', pattern: "*.id2details" }

    input:
    path(annotation)
    val(gtf_attr_type)
    val(gtf_feature_type_of_attr_type)

    output:
    path("${annotation.baseName}.id2details")

    shell:
    '''
    #!/usr/bin/env python3
    import sys
    target_id_set = set()
    with open("!{annotation}", 'r') as gtf, open("!{annotation.baseName}.id2details", 'a') as out:
        for line in gtf:
            if not line.startswith('#'):
                split_line = line.split('\\t')
                if split_line[2] == '!{gtf_feature_type_of_attr_type}' or split_line[2] == 'pseudogene':
                    if '!{gtf_attr_type}' not in line:
                        sys.exit(f"ERROR: No '!{gtf_attr_type}' found in a '!{gtf_feature_type_of_attr_type}'-type line:\\n{line}\\nCheck your an
                    target_id = line.split('!{gtf_attr_type}')[1].split(';')[0].replace('"', '').strip()
                    if target_id not in target_id_set:
                        desc = split_line[8]
                        chr = split_line[0]
                        start = split_line[3]
                        stop = split_line[4]
                        strand = split_line[6]
                        if '!{gtf_attr_type}' == 'transcript_id' and 'transcript_name' in desc:
                            target_name = desc.split('transcript_name')[1].split(';')[0].replace('"','').strip()
                        else:
                            if 'gene_name' in desc:
                                target_name = desc.split('gene_name')[1].split(';')[0].replace('"','').strip()
                            else:
                                target_name = target_id
                        if target_name == 'NA':
                            target_name = target_id
                        if '!{gtf_attr_type}' == 'transcript_id' and 'transcript_biotype' in desc:
                            target_biotype = desc.split('transcript_biotype')[1].split(';')[0].replace('"','').strip()
                        else:
                            if 'gene_biotype' in desc:
                                target_biotype = desc.split('gene_biotype')[1].split(';')[0].replace('"','').strip()
                            else:
                                target_biotype = 'NA'
                        target_id_set.add(target_id)
                        out.write('\\t'.join([target_id, target_name, target_biotype, chr, start, stop, strand, desc.rstrip()]) + '\\n')
    '''
```





RNAflow: An Effective and Simple RNA-Seq Differential Gene Expression Pipeline Using Nextflow

by Marie Lataretu [1] and Martin Hölzer [2,*]

[1] RNA Bioinformatics and High-Throughput Analysis, Friedrich Schiller University Jena, Leutragraben 1, 07743 Jena, Germany

[2] Methodology and Research Infrastructure, MF1 Bioinformatics, Robert Koch Institute, Nordufer 20, 13353 Berlin, Germany

[*] Author to whom correspondence should be addressed.

Genes **2020**, *11*(12), 1487; https://doi.org/10.3390/genes11121487

Submission received: 25 October 2020 / Revised: 4 December 2020 / Accepted: 7 December 2020 / Published: 10 December 2020

(This article belongs to the Special Issue Algorithms and Workflows in RNA Bioinformatics)

Abstract

RNA-Seq enables the identification and quantification of RNA molecules, often with the aim of detecting differentially expressed genes (DEGs). Although RNA-Seq evolved into a standard technique, there is no universal gold standard for these data's computational analysis. On top of that, previous studies proved the irreproducibility of RNA-Seq studies. Here, we present a portable, scalable, and parallelizable Nextflow RNA-Seq pipeline to detect DEGs, which assures a high level of reproducibility. The pipeline automatically takes care of common pitfalls, such as ribosomal RNA removal and low abundance gene filtering. Apart from various visualizations for the DEG results, we incorporated downstream pathway analysis for common species as *Homo sapiens* and *Mus musculus*. We evaluated the DEG detection functionality while using qRT-PCR data serving as a reference and observed a very high correlation of the logarithmized gene expression fold changes.

Keywords: RNA-Seq; workflow; Nextflow pipeline; differential gene expression analysis

1. Introduction

More than a decade ago, the possibility to sequence the transcriptome (RNA-Seq) of any given species, starting from small bacteria, such as *Helicobacter pylori* [1], opened up completely new ways to analyze gene expression and obtain never before possible insights into regulatory mechanisms on an incredibly large scale. Starting from short, single-end reads and few samples per study, we now have access to continuously growing and large transcriptomic data sets of high sequencing quality [2]. A standard and widely distributed computational task comprises the processing of RNA-Seq data to identify differentially expressed features, mainly genes (DEGs), between varying conditions, such as different tissue types or control vs. treated samples. Although being frequently performed nowadays, setting up the necessary computational steps to perform an RNA-Seq-based DEG study is still time-consuming and prone to errors, which ultimately leads to results that are difficult to reproduce [3].

Code: GitHub - data science

Graph: workflow structure

Text: scientific papers

BioFlow-Insight

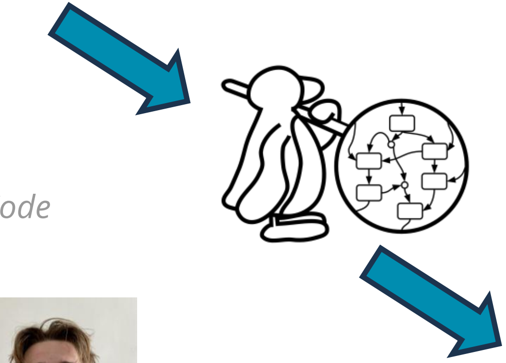
Aim

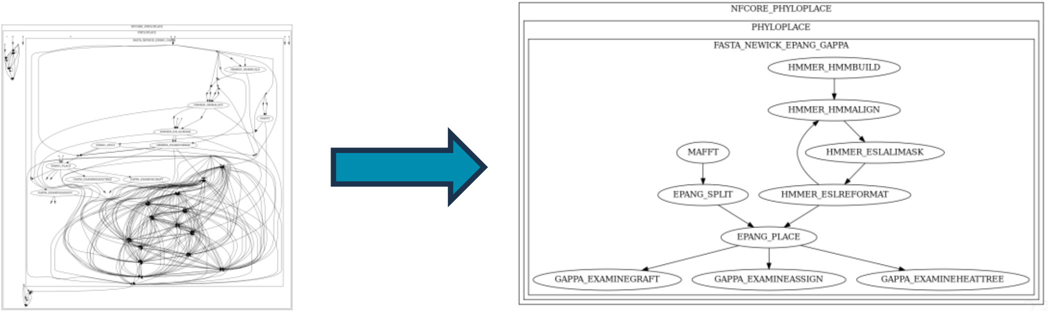
Provide an overview of the main step of a workflow

Coherent with the code…

nextflow

Workflow Code





George **Marchment**

BioFlow-Insight

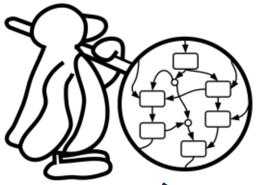
Aim

Provide an overview of the main step of a workflow

Coherent with the code…

nextflow

Workflow Code



George **Marchment**

Functionalities

Analyses the code of Nextflow workflows

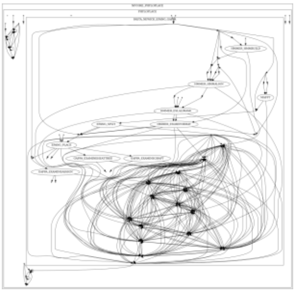
Generates visual graphs at different **simplification** levels depicting the workflow's structure

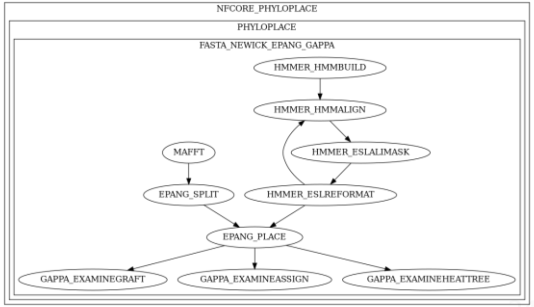
Detects errors in the workflow code

Open source: command line or web service

https://bioflow-insight.pasteur.cloud/

NAR Genomics & Bioinformatics

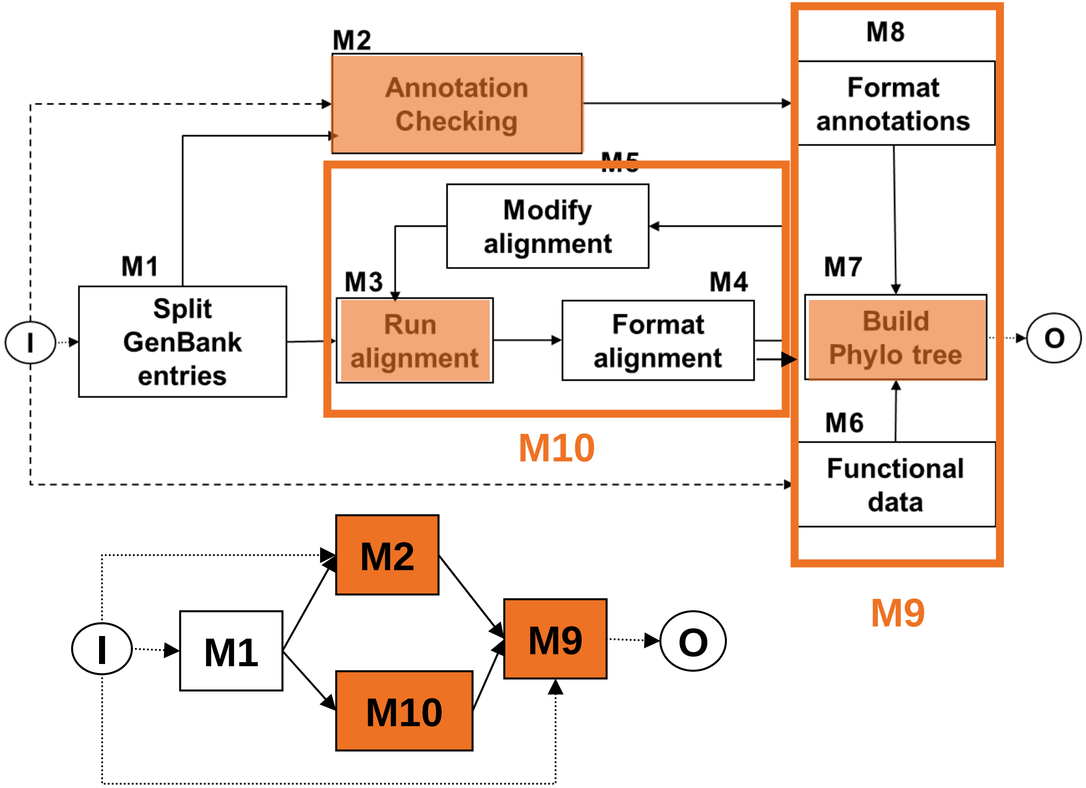




Simplifying the workflow graph

Choose **relevant modules**

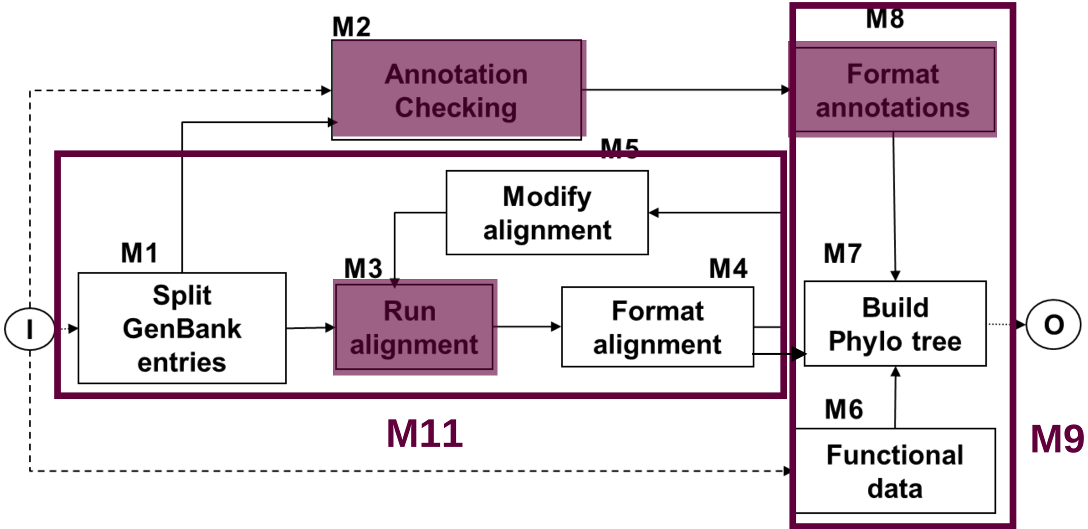
Each **composite** module is constructed around one **relevant module**



The composite module takes the meaning of the relevant module it contains

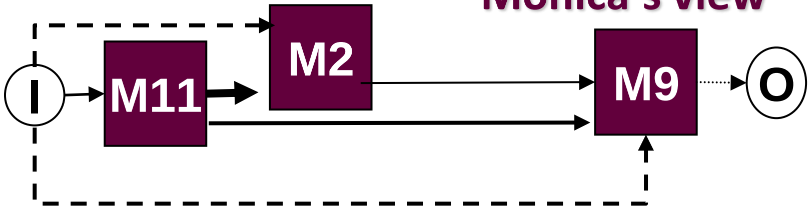
Grouping may be error-prone!

Grouping should preserve the relationships between relevant modules



Monica's view

The *annotation checking* module does not need input from the *run alignment* module!



ZOOM*UserViews in NextFlow Workflows

Formalization of the set of properties to be preserved

Property 1: Given Gw and R ⊆ N relevant modules, U is well-formed iff every composite module in U contains at most one element of R.

Property 2: A user view U preserves dataflow iff every edge in Gw that induces an edge on an nr-path from C(r) to C(r’) in U(Gw) lies on an nr-path r to r’ in Gw.

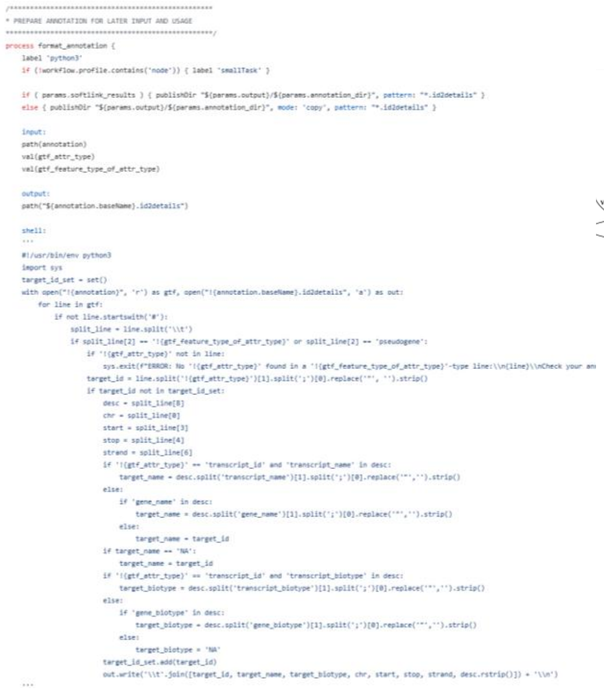
Property 3: A user view U is complete w.r.t dataflow iff for every edge e on an nr-path from r to r’ in Gw that induces an edge e’ in U(Gw), e’ lies on an nr-path from C(r) to C(r’).

Theorem *ZOOM* is a polynomial-time which *preserves Properties 1- 3 and produces a minimal user view*

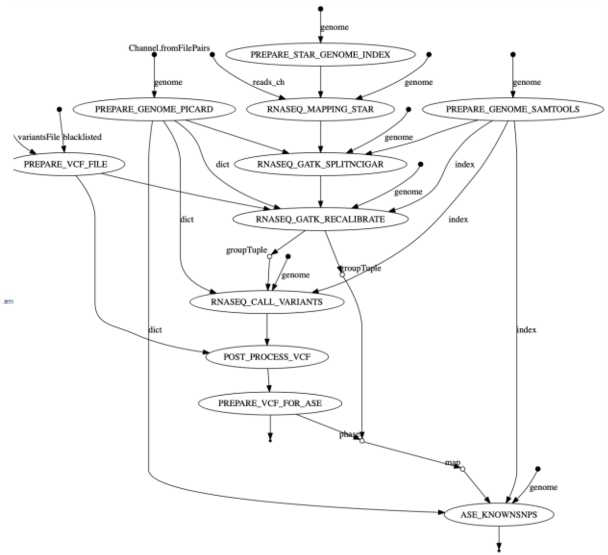
Implementation of ZOOM

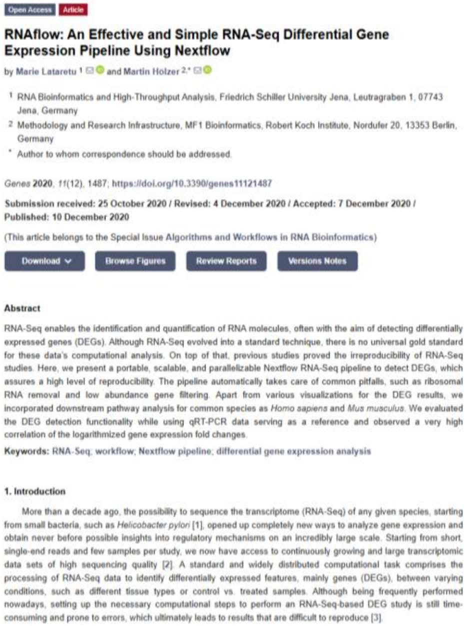
New! Rewritting the workflow code to implement user views

Exploiting the 3 forms of the workflow



```
/****************************************************
* PREPARE ANNOTATION FOR LATER INPUT AND USAGE
****************************************************/
process format_annotation {
    label 'python3'
    if (!workflow.profile.contains('node')) { label 'smallTask' }

    if ( params.softlink_results ) { publishDir "${params.output}/${params.annotation_dir}", pattern: "*.id2details" }
    else { publishDir "${params.output}/${params.annotation_dir}", mode: 'copy', pattern: "*.id2details" }

    input:
    path(annotation)
    val(gtf_attr_type)
    val(gtf_feature_type_of_attr_type)

    output:
    path("${annotation.baseName}.id2details")

    shell:
    '''
    #!/usr/bin/env python3
    import sys
    target_id_set = set()
    with open("!{annotation}", 'r') as gtf, open("!{annotation.baseName}.id2details", 'a') as out:
        for line in gtf:
            if not line.startswith('#'):
                split_line = line.split('\\t')
                if split_line[2] == '!{gtf_feature_type_of_attr_type}' or split_line[2] == 'pseudogene':
                    if '!{gtf_attr_type}' not in line:
                        sys.exit(f"ERROR: No '!{gtf_attr_type}' found in a '!{gtf_feature_type_of_attr_type}'-type line:\\n{line}\\nCheck your an
                    target_id = line.split('!{gtf_attr_type}')[1].split(';')[0].replace('"', '').strip()
                    if target_id not in target_id_set:
                        desc = split_line[8]
                        chr = split_line[0]
                        start = split_line[3]
                        stop = split_line[4]
                        strand = split_line[6]
                        if '!{gtf_attr_type}' == 'transcript_id' and 'transcript_name' in desc:
                            target_name = desc.split('transcript_name')[1].split(';')[0].replace('"','').strip()
                        else:
                            if 'gene_name' in desc:
                                target_name = desc.split('gene_name')[1].split(';')[0].replace('"','').strip()
                            else:
                                target_name = target_id
                        if target_name == 'NA':
                            target_name = target_id
                        if '!{gtf_attr_type}' == 'transcript_id' and 'transcript_biotype' in desc:
                            target_biotype = desc.split('transcript_biotype')[1].split(';')[0].replace('"','').strip()
                        else:
                            if 'gene_biotype' in desc:
                                target_biotype = desc.split('gene_biotype')[1].split(';')[0].replace('"','').strip()
                            else:
                                target_biotype = 'NA'
                        target_id_set.add(target_id)
                        out.write('\\t'.join([target_id, target_name, target_biotype, chr, start, stop, strand, desc.rstrip()]) + '\\n')
    '''
```





RNAflow: An Effective and Simple RNA-Seq Differential Gene Expression Pipeline Using Nextflow

by Marie Lataretu [1] and Martin Hölzer [2,*]

[1] RNA Bioinformatics and High-Throughput Analysis, Friedrich Schiller University Jena, Leutragraben 1, 07743 Jena, Germany

[2] Methodology and Research Infrastructure, MF1 Bioinformatics, Robert Koch Institute, Nordufer 20, 13353 Berlin, Germany

[*] Author to whom correspondence should be addressed.

Genes **2020**, *11*(12), 1487; https://doi.org/10.3390/genes11121487

Submission received: 25 October 2020 / Revised: 4 December 2020 / Accepted: 7 December 2020 / Published: 10 December 2020

(This article belongs to the Special Issue Algorithms and Workflows in RNA Bioinformatics)

Abstract

RNA-Seq enables the identification and quantification of RNA molecules, often with the aim of detecting differentially expressed genes (DEGs). Although RNA-Seq evolved into a standard technique, there is no universal gold standard for these data's computational analysis. On top of that, previous studies proved the irreproducibility of RNA-Seq studies. Here, we present a portable, scalable, and parallelizable Nextflow RNA-Seq pipeline to detect DEGs, which assures a high level of reproducibility. The pipeline automatically takes care of common pitfalls, such as ribosomal RNA removal and low abundance gene filtering. Apart from various visualizations for the DEG results, we incorporated downstream pathway analysis for common species as *Homo sapiens* and *Mus musculus*. We evaluated the DEG detection functionality while using qRT-PCR data serving as a reference and observed a very high correlation of the logarithmized gene expression fold changes.

Keywords: RNA-Seq; workflow; Nextflow pipeline; differential gene expression analysis

1. Introduction

More than a decade ago, the possibility to sequence the transcriptome (RNA-Seq) of any given species, starting from small bacteria, such as *Helicobacter pylori* [1], opened up completely new ways to analyze gene expression and obtain never before possible insights into regulatory mechanisms on an incredibly large scale. Starting from short, single-end reads and few samples per study, we now have access to continuously growing and large transcriptomic data sets of high sequencing quality [2]. A standard and widely distributed computational task comprises the processing of RNA-Seq data to identify differentially expressed features, mainly genes (DEGs), between varying conditions, such as different tissue types or control vs. treated samples. Although being frequently performed nowadays, setting up the necessary computational steps to perform an RNA-Seq-based DEG study is still time-consuming and prone to errors, which ultimately leads to results that are difficult to reproduce [3].

Code: GitHub - data science

Graph: workflow structure

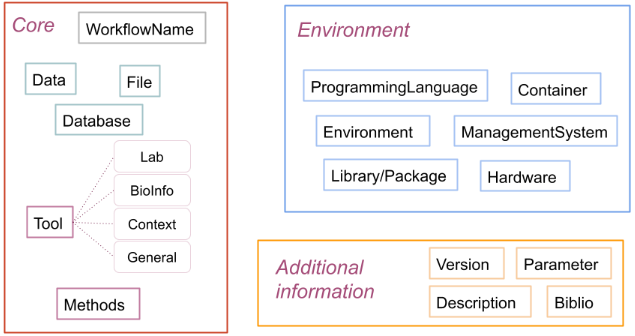
Text: scientific papers

Entity extraction from scientific papers

Low-resource extraction task

- Bioinformatics workflows underrepresented in NLP
- No Corpus available

Several strategies tested



Core

WorkflowName

Data

File

Database

Tool

Lab

BioInfo

Context

General

Methods

Environment

ProgrammingLanguage

Container

Environment

ManagementSystem

Library/Package

Hardware

Additional information

Version

Parameter

Description

Biblio

Schema of entities representing key entities related to workflow

Annotated corpus of 52 full papers describing Snakemake and Nextflow workflows

- 7 annotators - Inter annot. Agreement (0.70)

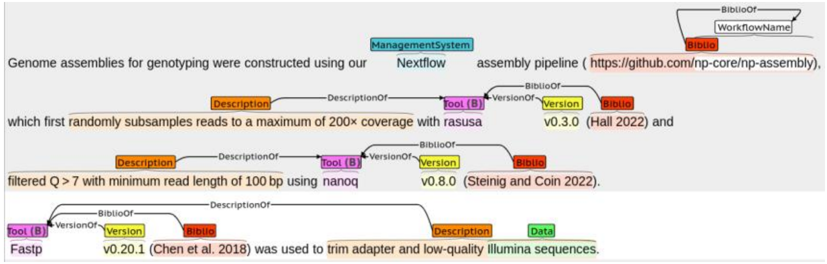
Neuronal approach biLSTM-CRF

- Nlstruct python lib
- SciBert & BioBert

F1-score ~0.70 all entities (0.77 on Tools)

Accepted at IDA 2025

IDA 2025



Genome assemblies for genotyping were constructed using our Nextflow assembly pipeline (https://github.com/np-core/np-assembly), which first randomly subsamples reads to a maximum of 200× coverage with rasusa v0.3.0 (Hall 2022) and filtered Q > 7 with minimum read length of 100 bp using nanoq v0.8.0 (Steinig and Coin 2022). Fastp v0.20.1 (Chen et al. 2018) was used to trim adapter and low-quality Illumina sequences.

Clémence **Sebe**

Wrap up - Next steps

Lack of reproducibility hurts **cumulative science**

We are still living the **reproducibility crisis**

Several technical solutions exist to help redo/reexecute

Challenges lie at **the reuse level**: repurposing workflow analyses, adaptating to own needs

ShareFAIR ambitions to provide a **proof-of-concept of workflow sharing** by providing a **reuse platform**

Current work on

Coupling **workflow code & workflow papers** (NLP-code)

Abstracting workflow **graph structures** (code-graph)

FR RR

RR FR FR Reproducible Research network

April 3-4 2025
Lyon

Thanks!

BioFlow-Insight: Leveraging Graph representations to Facilitate the Reuse of Bioinformatics Workflows
Scientific Workflows enable Reproducible analyses
Many Workflows available Online
Steep Learning Curve
Can be difficult to Understand
Goldmine for Reuse
Graphical Representation
BioFlow-Insight
Error Detection
Usable from Website or Command Line
Facilitates the Reuse of Workflows
From Headache to High-Five! We are trying to make reproducibility in Bioinformatics easier
An initial exploration into navigating fine-grained provenance and using its precious information
Background
Results
Methods
BioToFlow : le premier corpus sur les workflows bioinformatiques
Corpus BioToFlow
Workflow bioinformatique
Expérimentations
19
PARIS NEW-YORK VITRÉ TOKYO LONDRE