HPC Pipelines for Reproducibility and Profit

Alastair Droop, 2024-06-12

Reproducibility & Scientific Software

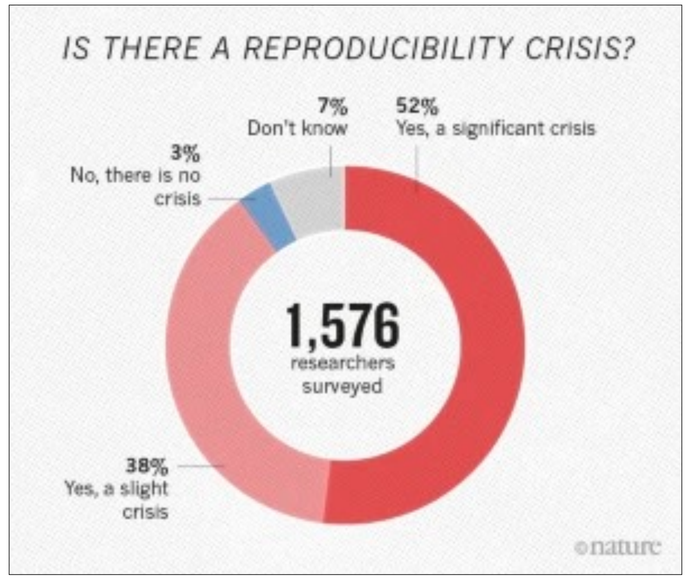
(Bio)science is suffering from a major reproducibility crisis

Published results frequently can not be reproduced

A major aspect of this is a lack of reproducibility in scientific software

Many aspects to this problem

- (Modern) scientific code is complicated & complex
- Not enough time or resources to “do software engineering properly”
- Not enough training
- Inappropriate tools



Baker, M. “1,500 scientists lift the lid on reproducibility”. *Nature* **533**, 452–454 (2016). https://doi.org/10.1038/533452a

The FAIR Principles in Scientific Computing

Findable

- Users can find (specific versions of) the software using a unique and persistent identifier

Accessible

- Software can be accessed and installed using standard tools

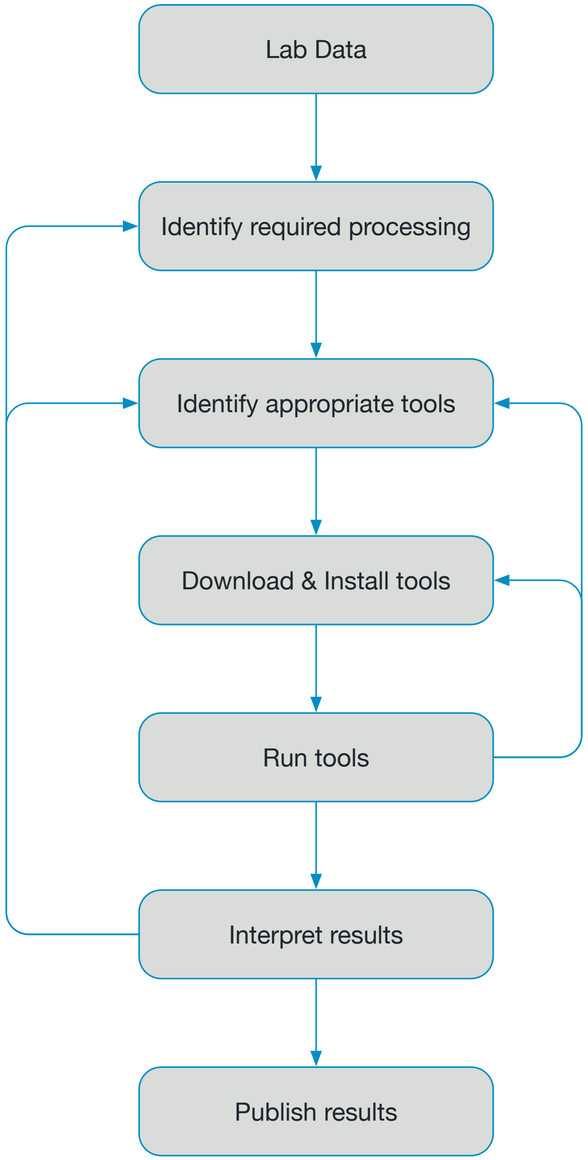
Interoperable

- Software adheres to domain-relevant data standards

Reusable

- Software can be run by other users for their specific needs

Current Practice in Research Code



Individual labs identify & install published code to run analyses

Little understanding of best practice or code reproducibility

Multiple installations of each tool with different versions

- And no record of version numbers

Usually, data can not be re-generated

Dependencies

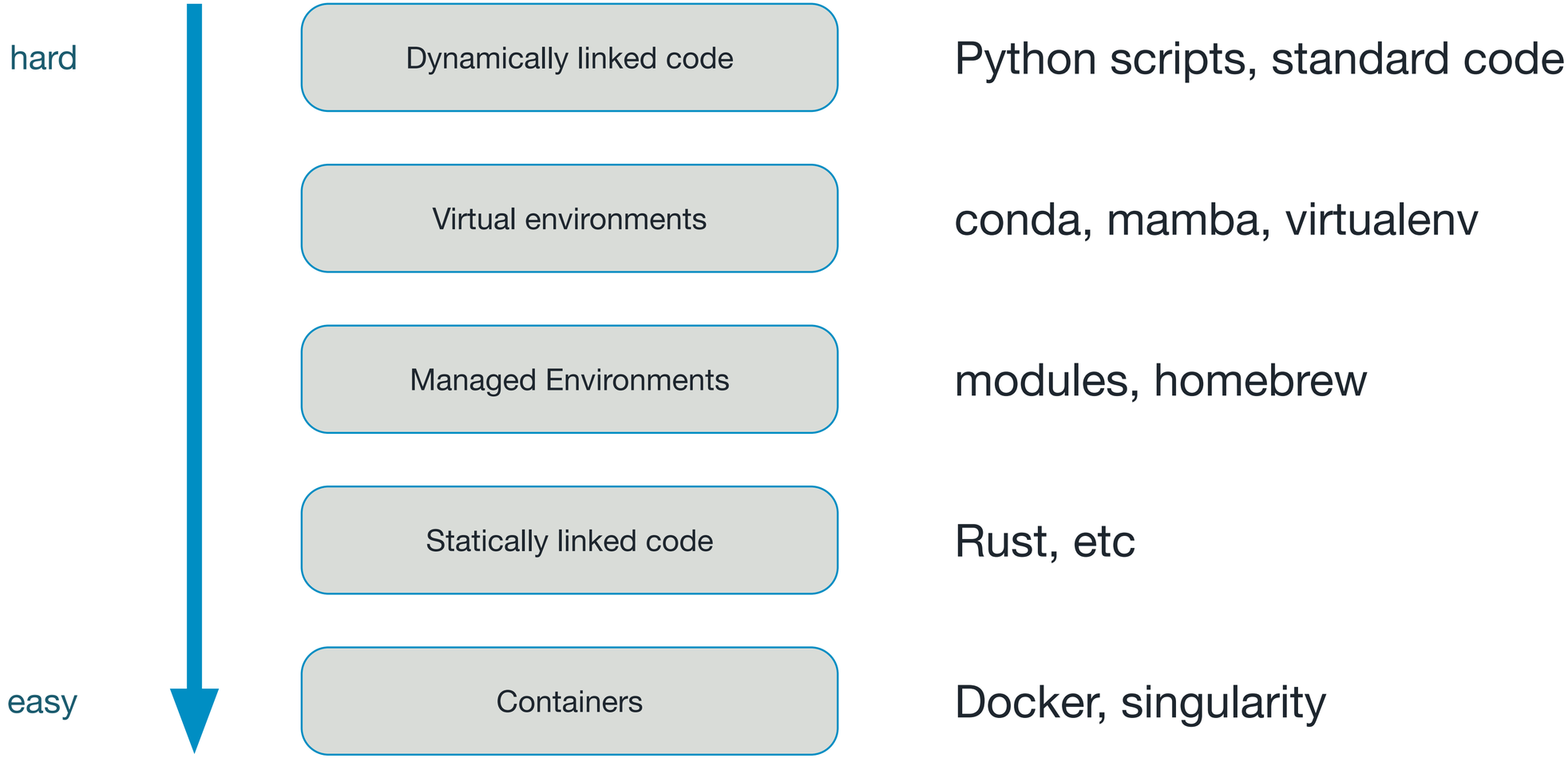
A fundamental problem of interpreted languages is runtime dependency management

Code needs to locate (at runtime) dependency code and run it

Updating some code that a program depends on can change its behaviour

Many ways to address this problem

Containerisation



What's Wrong with Python?

Python is interpreted

- The runtime is slow, and needs to perform garbage collection periodically
- The runtime makes writing multi-threaded code hard
- The poor runtime can't see all the code at once, so can't perform (full) static analysis

Python is dynamically typed

- Static analysis is very hard / impossible
- Debugging is harder

Python is written in C

- Hard to find deep errors

Python's dependency stack is not well defined

UNIVERSITY of York

How Should you Install a Python Package?

`setuptools`, **`pip`**, **`venv`**, **`wheel`**, **`twine`**, `pip-tools`, `virtualenvwrapper`, **`pipx`**, `conda`, **`pipenv`**, `poetry`, **`flit`**, **`hatch`**, `pdm`

Most of these are to a greater or lesser degree incompatible

- *Which one do you pick?*
- What happens if you need to install a pipeline with tools that are packaged in an incompatible way?

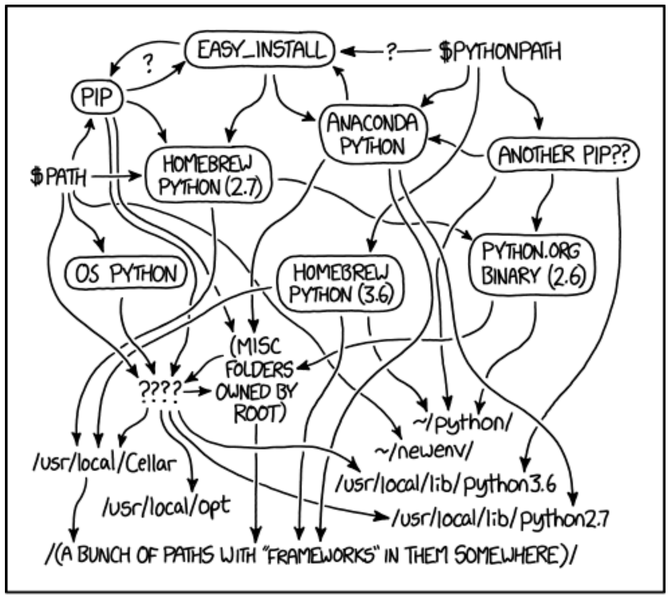
Virtual Environments are Difficult

We often need virtual environments to make R and Python software work

These are directories of packages that are loaded on demand

Virtual Environments are:

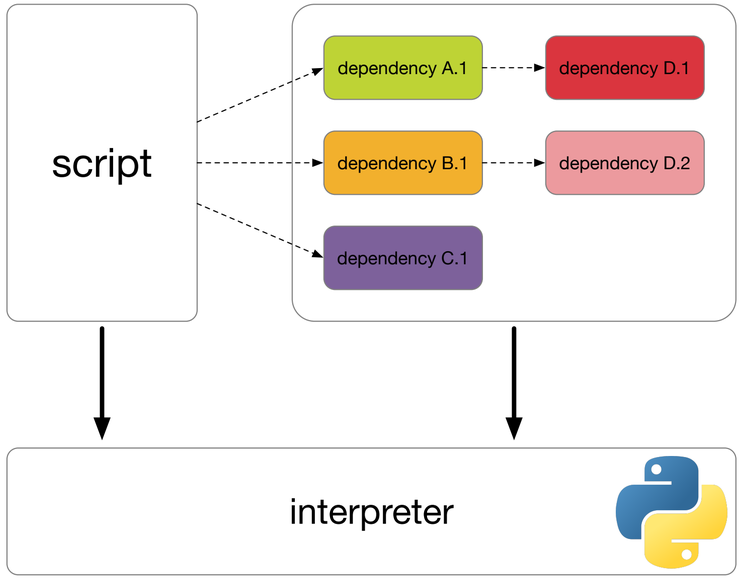
- Specific to a Python version
- Trivially updatable (and this is *really bad*)
- Fragile
- Often bloated
- Surprisingly difficult to accurately reproduce
- Difficult for users to set up



MY PYTHON ENVIRONMENT HAS BECOME SO DEGRADED THAT MY LAPTOP HAS BEEN DECLARED A SUPERFUND SITE.

https://xkcd.com/1987

Interpreted Languages

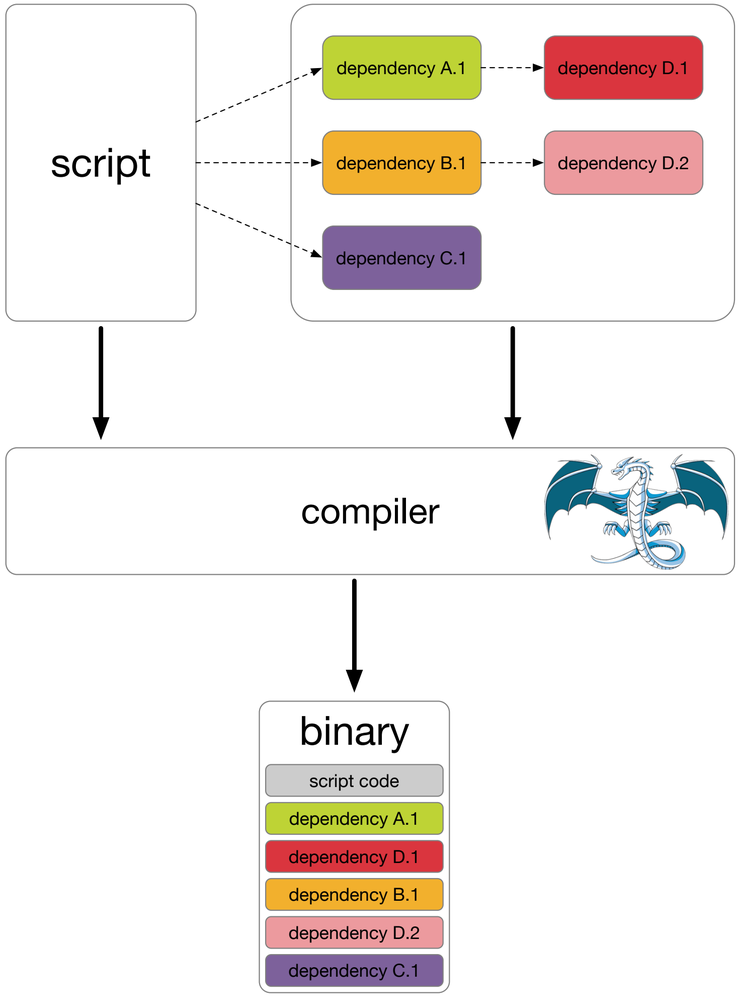


Interpreted languages are executed by an interpreter at runtime, which finds and links dependencies on the fly

+ Code can be almost instantly run (no compilation step)
+ Code can be trivially modified
+ Base script is small
+ Scripts can be platform-agnostic

– Dependencies are not included in the code
– Installation can be very complex
– Interpreter needs to be running, so often slower

Statically Compiled Languages



Statically-compiled code is executed by the system at runtime, and contains all of its dependencies compiled into a single binary

+ Code does not rely on external packages that can get lost
+ Compiler can run comprehensive checks on the code
+ Installation can be extremely easy
+ No runtime interpreter required
+ Small runtime overhead

– Compilation can be slow
– Executables can be quite large
– Compiled binaries are platform-specific

Benefits of (Static) Compilation

If we compile our code, the compiler gets a chance to see all the code at once

This allows the compiler to perform rigorous error checking

Rust takes this much further than most languages

+ Many whole classes of bug are no longer possible
+ The need for garbage collection is removed
+ Strict data ownership can be observed, allowing for highly parallel code

Containerisation

The purest form of method of containerising software is to bundle the tool and OS into a single box

Tools such as Docker & Apptainer do this

+ All dependencies are built into a single image
+ Installation is trivial
+ The exact software built into a container is documented
+ Containers can be hosted publicly (for example on quay.io)
+ Runtime use is standardised

– … none of the above points are really that simple
– Containers can be huge
– Easy stuff is trivial, but hard stuff gets *really* hard

Containerisation

We build & ship containers using Docker, but run on HPC using Apptainer

"Best of both Worlds"

Requires root privileges

Easy to run on Linux, Mac, Windows, laptops, HPC

Well established & standard runtime & formats

Excellent online container repositories

Does not require root privileges

Relatively complex to run off HPC

Still relatively niche

Containerisation

If necessary, build & test image locally (on laptop) by writing a Dockerfile

Once complete & tested, push Dockerfile to dockerhub

Pull container from dockerhub via apptainer onto Viking

Containerisation

```
FROM debian
LABEL image.author.name "Alastair Droop"
LABEL image.author.email "alastair.droop@york.ac.uk"

USER root
RUN apt-get -y update &&
    apt-get -y install --fix-missing &&
    apt-get install -y rng-tools-debian ent git wget gcc g++ make libbz2-dev libdivsufsort-dev libjsoncpp-dev libssl-dev libmpfr-dev

WORKDIR /root
RUN mkdir github
WORKDIR /root/github
RUN git clone https://github.com/blep/TestU01.git
WORKDIR /root/github/TestU01
RUN rm config.guess config.sub &&
    wget http://savannah.gnu.org/cgi-bin/viewcvs/*checkout*/config/config/config.guess &&
    wget http://savannah.gnu.org/cgi-bin/viewcvs/*checkout*/config/config/config.sub &&
    chmod a+x config.guess config.sub &&
    ./configure &&
    make &&
    make install
ENV PATH="/root/github/NIST-Statistical-Test-Suite/sts:$PATH"

RUN mkdir /root/github/alpharabbit
WORKDIR /root/github/alpharabbit
ADD alpharabbit.c alpharabbit.c
ENV LD_LIBRARY_PATH=/usr/local/lib:${LD_LIBRARY_PATH}
ENV LIBRARY_PATH=/usr/local/lib:${LIBRARY_PATH}
ENV C_INCLUDE_PATH=/usr/local/include:${C_INCLUDE_PATH}
RUN gcc alpharabbit.c -o alpharabbit -ltestu01 -lprobdist -lmylib -lm
ENV PATH="/root/github/alpharabbit:$PATH"

ENV DATA="/mnt/data"

ENV HOME=/root
```

Pipelines

Even if each script is perfectly containerised, we still need to record the steps performed

Pipelines simplify this

- Pipelines are conceptual workflows that manage & record the flow of data through a defined set of steps
- Provide the logic defining processes to be performed, as well as a record of what was done

A good pipeline system provides

- an *exact* log of what was done to your data; and
- A simple way to re-run the same steps

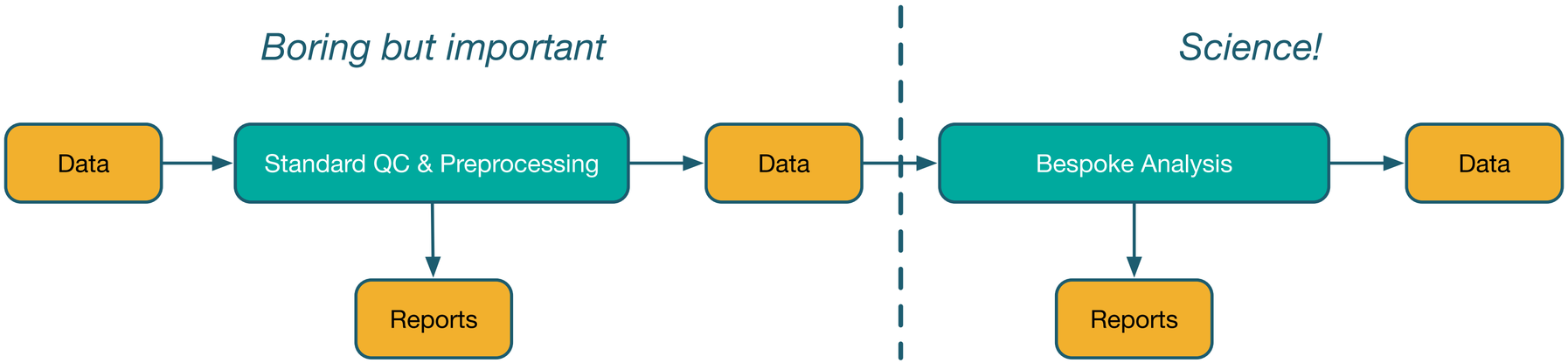
To be most effective, we need to ensure we can standardise as much as possible

Multiple Attempts to build Pipelines

CWL, **Snakemake**, Galaxy, **Nextflow**, WDL, GNU make, bpipe, Cgpipe, Tfpipe, BigDataScript, Anduril, Ruffus, Loom

We actively use Snakemake & Nextflow

- Snakemake allows us to easily link multiple scripts together for once-off (or infrequent) use
- Nextflow allows us to run complex pipelines at high scale



Nextflow

We use Nexflow for our HPC pipeline work

+ Vast community effort
+ Industry buy-in
+ Many executors supported (including Slurm)
+ Supports containers
+ Extremely flexible

– Steep learning curve
– Groovy? *Really*?
– Easy stuff is trivial, but hard stuff gets *really* hard

NF Core

The nf-core project hosts a large and growing set of nextflow bioinformatics pipelines

+ Vast community effort
+ Standardised format & underlying logic
+ Pipeline definition schemata
+ Extensive support & documentation
+ Many common biology processing tasks covered
+ Submission is very simple on HPC

– Steep learning curve (again)

Nf-core Pipelines

rnaseq ✓ ☆ 796

RNA sequencing analysis pipeline using STAR, RSEM, HISAT2 or Salmon with gene/isoform counts and extensive quality control.

rna rna-seq

3.14.0 released 5 months ago

sarek ✓ ☆ 346

Analysis pipeline to detect germline or somatic variants (pre-processing, variant calling and annotation) from WGS / targeted sequencing

annotation cancer gatk4 genomics germline pre-processing somatic target-panels variant-calling whole-exome-sequencing whole-genome-sequencing

3.4.2 released about 1 month ago

mag ✓ ☆ 187

Assembly and binning of metagenomes

annotation assembly binning long-read-sequencing metagenomes metagenomics nanopore nanopore-sequencing

3.0.1 released 2 days ago

chipseq ✓ ☆ 175

ChIP-seq peak-calling, QC and differential analysis pipeline.

chip chip-seq chromatin-immunoprecipitation macs2 peak-calling

2.0.0 released over 1 year ago

scrnaseq ✓ ☆ 174

A single-cell RNAseq pipeline for 10X genomics data

10x-genomics 10xgenomics alevin bustools cellranger kallisto rna-seq single-cell star-solo

2.6.0 released about 1 month ago

atacseq ✓ ☆ 164

ATAC-seq peak-calling and QC analysis pipeline

atac-seq chromatin-accessibiity

2.1.2 released 10 months ago

ampliseq ✓ ☆ 161

Amplicon sequencing analysis workflow using DADA2 and QIIME2

16s 18s amplicon-sequencing edna illumina iontorrent its metabarcoding metagenomics microbiome pacbio qiime2 rrna taxonomic-classification taxonomic-profiling

2.9.0 released 2 months ago

nanoseq ✓ ☆ 150

Nanopore demultiplexing, QC and alignment pipeline

alignment demultiplexing nanopore qc

3.1.0 released over 1 year ago

methylseq ✓ ☆ 135

Methylation (Bisulfite-Sequencing) analysis pipeline using Bismark or bwa-meth + MethylDackel

bisulfite-sequencing dna-methylation em-seq epigenome epigenomics methyl-seq pbat rrbs

2.6.0 released 5 months ago

rnafusion ✓ ☆ 130

RNA-seq analysis pipeline for detection of gene-fusions

fusion fusion-genes gene-fusion rna rna-seq

3.0.2 released 2 months ago

fetchngs ✓ ☆ 126

Pipeline to fetch metadata and raw FastQ files from public databases

ddbj download ena fastq geo sra synapse

1.12.0 released 3 months ago

eager ✓ ☆ 124

A fully reproducible and state-of-the-art ancient DNA analysis pipeline

adna ancient-dna-analysis ancientdna genome metagenomics pathogen-genomics population-genetics

2.5.1 released 4 months ago

viralrecon ✓ ☆ 110

Assembly and intrahost/low-frequency variant calling for viral samples

amplicon artic assembly covid-19 covid19 illumina long-read-sequencing metagenomics nanopore ont oxford-nanopore sars-cov-2 variant-calling viral virus

2.6.0 released about 1 year ago

taxprofiler ✓ ☆ 100

Highly parallelised multi-taxonomic profiling of shotgun short- and long-read metagenomic data

classification illumina long-reads metagenomics microbiome nanopore pathogen profiling shotgun taxonomic-classification taxonomic-profiling

1.1.7 released about 1 month ago

hic ✓ ☆ 79

Analysis of Chromosome Conformation Capture data (Hi-C)

chromosome-conformation-capture hi-c

2.1.0 released about 1 year ago

raredisease ✓ ☆ 74

Call and score variants from WGS/WES of rare disease patients.

diagnostics rare-disease snv structural-variants variant-annotation variant-calling wes wgs

2.1.0 released 14 days ago

smrnaseq ✓ ☆ 69

A small-RNA sequencing analysis pipeline

small-rna smrna-seq

2.3.1 released about 2 months ago

cutandrun ✓ ☆ 66

Analysis pipeline for CUT&RUN and CUT&TAG experiments that includes QC, support for spike-ins, IgG controls, peak calling and downstream analysis.

cutandrun cutandrun-seq cutandtag cutandtag-seq

3.2.2 released 4 months ago

funcscan ✓ ☆ 60

(Meta-)genome screening for functional and natural product gene sequences

amp amr antibiotic-resistance antimicrobial-peptides antimicrobial-resistance-genes arg assembly bgc biosynthetic-gene-clusters contigs function metagenomics natural-products screening secondary-metabolites

1.1.5 released 3 months ago

bacass ✓ ☆ 55

Simple bacterial assembly and annotation pipeline

assembly bacterial-genomes denovo denovo-assembly genome-assembly hybrid-assembly nanopore nanopore-sequencing

2.2.0 released about 2 months ago

pangenome ✓ ☆ 54

Renders a collection of sequences into a pangenome graph.

pangenome

1.1.2 released 3 months ago

hlatyping ✓ ☆ 52

Precision HLA typing from next-generation sequencing data

dna hla hla-typing immunology optitype personalized-medicine rna

2.0.0 released over 1 year ago

bactmap ✓ ☆ 48

A mapping-based pipeline for creating a phylogeny from bacterial whole genome sequences

bacteria bacterial bacterial-genome-analysis genomics mapping phylogeny tree

1.0.0 released almost 3 years ago

differentialabundance ✓ ☆ 46

Differential abundance analysis for feature/ observation matrices from platforms such as RNA-seq

atac-seq chip-seq deseq2 differential-abundance differential-expression gsea limma microarray rna-seq shiny

1.5.0 released about 1 month ago

airrflow ✓ ☆ 44

B-cell and T-cell Adaptive Immune Receptor Repertoire (AIRR) sequencing analysis pipeline using the Immcantation framework

airr b-cell immcantation immunorepertoire repseq

proteinfold ✓ ☆ 41

Protein 3D structure prediction pipeline

alphafold2 protein-fold-prediction protein-folding protein-sequences protein-structure

spatialvi ☆ 41

Pipeline for processing spatially-resolved gene counts with spatial coordinates and image data. Designed for 10x Genomics Visium transcriptomics.

10x-genomics 10xgenomics image-processing microscopy rna-seq single-cell spatial spatial-transcriptomics st transcriptomics visium

deepvariant ☆ 40

Please consider using/contributing to https://github.com/nf-core/sarek

deep-variant dna google variant-calling

circrna ☆ 40

circRNA quantification, differential expression analysis and miRNA target prediction of RNA-Seq data

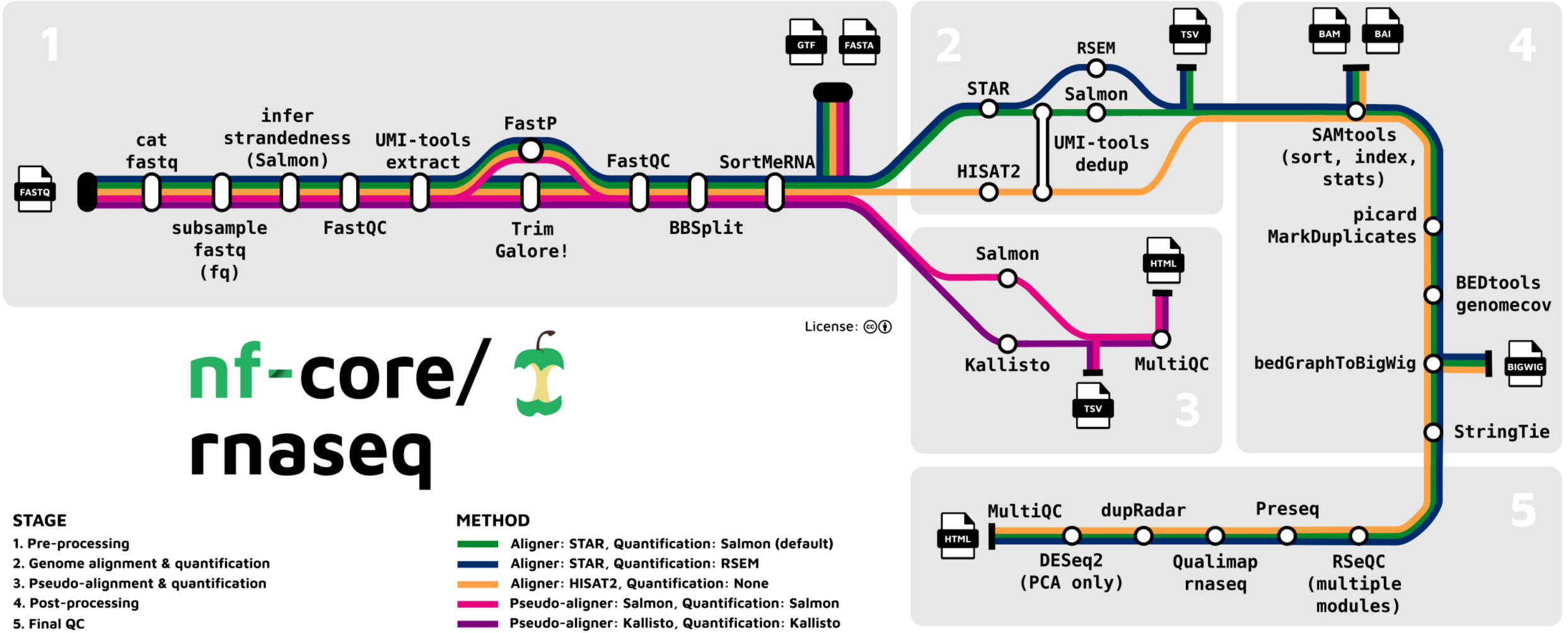
circrna circrna-pipeline circrna-prediction circular-rna genomics mirna mirna-targets ngs rna-seq

epitopeprediction ✓ ☆ 37

A bioinformatics best-practice analysis pipeline for epitope prediction and annotation

epitope epitope-prediction mhc-binding-prediction

The nf-core/rnaseq Pipeline



Seqera Labs

nf-core pipelines can be run “directly” on the HPC as sets of Slurm jobs

It is much easier to use a standardised portal to manage these We use Seqera Platform

+ Trivial submission
+ At-a-glance tracking
+ Simple run metrics
+ Simple logs & reports
+ Easy to migrate to other HPC platforms as well as Viking

– Paid service

RNASeq Example

Input Samples are provides as a CSV file:

sample	fastq_1	fastq_2	strandedness
Control.1	/mnt/scratch/projects/biol-tf-2018/data/SRR1272186_1.fastq.gz	/mnt/scratch/projects/biol-tf-2018/data/SRR1272186_2.fastq.gz	auto
Control.2	/mnt/scratch/projects/biol-tf-2018/data/SRR1272187_1.fastq.gz	/mnt/scratch/projects/biol-tf-2018/data/SRR1272187_2.fastq.gz	auto
Control.3	/mnt/scratch/projects/biol-tf-2018/data/SRR1272188_1.fastq.gz	/mnt/scratch/projects/biol-tf-2018/data/SRR1272188_2.fastq.gz	auto
Ischemic.1	/mnt/scratch/projects/biol-tf-2018/data/SRR1272189_1.fastq.gz	/mnt/scratch/projects/biol-tf-2018/data/SRR1272189_2.fastq.gz	auto
Idiopathic.2	/mnt/scratch/projects/biol-tf-2018/data/SRR1272190_1.fastq.gz	/mnt/scratch/projects/biol-tf-2018/data/SRR1272190_2.fastq.gz	auto
Idiopathic.3	/mnt/scratch/projects/biol-tf-2018/data/SRR1272191_1.fastq.gz	/mnt/scratch/projects/biol-tf-2018/data/SRR1272191_2.fastq.gz	auto

Run parameters provided as either JSON or via the GUI

Seqera Demo

(demo)

The TF Data Science Group

We're Part of the York University Bioscience Technology Facility

- Bioinformatics
- Machine learning & AI
- Mathematical / statistical techniques
- Modelling, simulation & visualisation
- Data reproducibility, security, anonymity
- Scientific software development

Teaching is increasingly important

- Specific techniques & skills
- Drop-in & "clinic" style advice
- Research computing skills
- Data handling
- Programming