Ceci n’est pas une pipeline

An Open Standard for Reproducible Genomics

Nebojsa Tijanic
Seven Bridges Genomics

NATURE METHODS | EDITORIAL

Software with impact

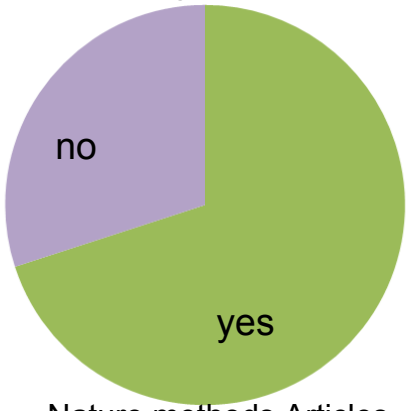
Nature Methods **11**, 211 (2014) | doi:10.1038/nmeth.2880
Published online 27 February 2014

The usefulness of computational methods can be improved by releasing code and designing software that supports reproducible research.

People are sharing code

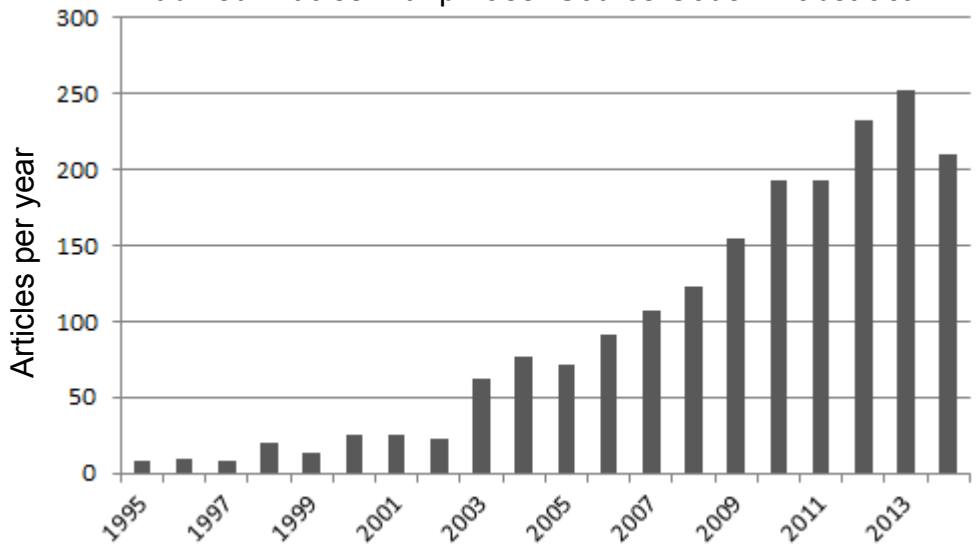
- Source Code
  - Post-publication as supplementary files
  - Public repositories (github, etc)

Supplementary Software files containing source code:



Nature methods Articles
Since 2007; n=133
doi:10.1038/nmeth.2880

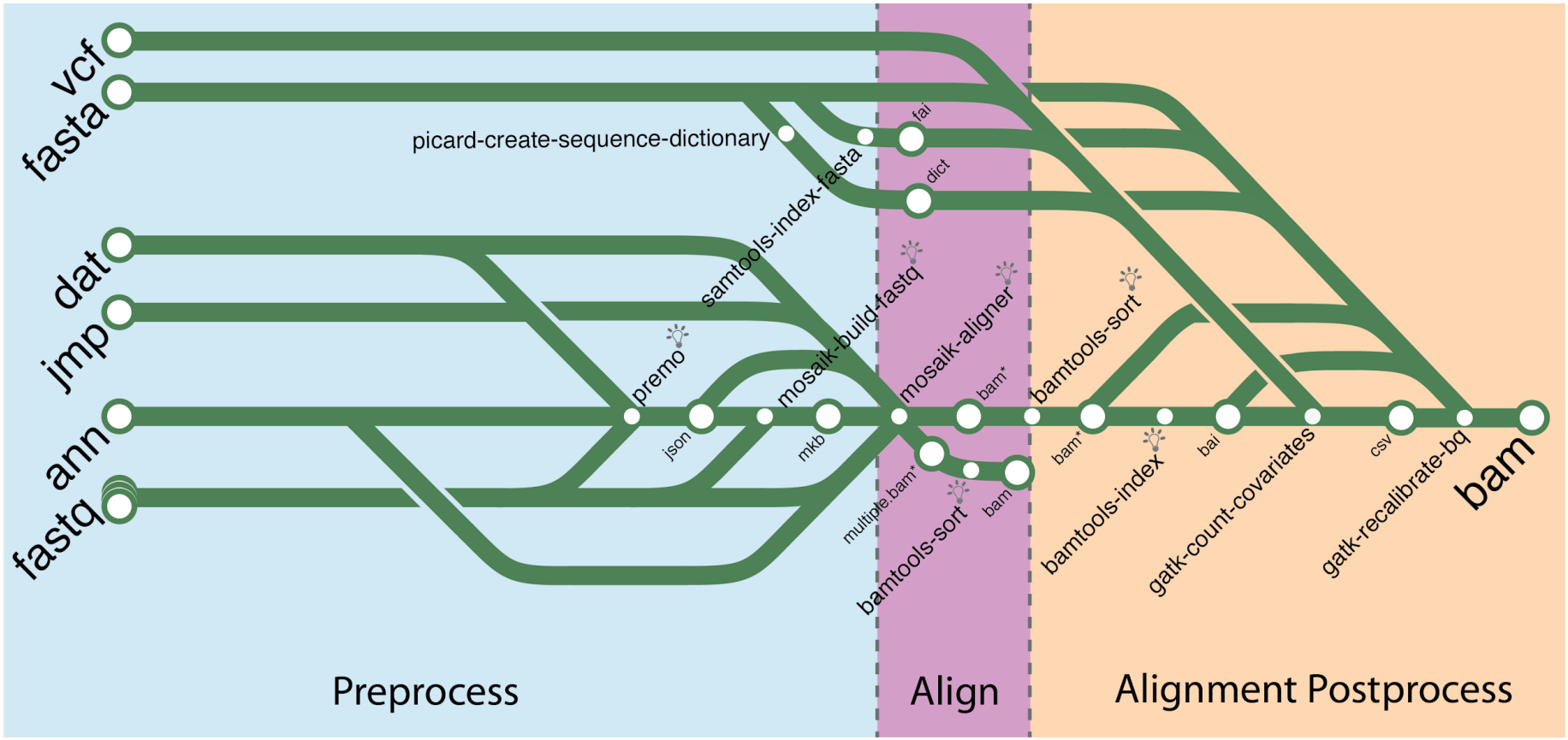
Pubmed Articles with phrase “Source Code” in abstract



accessed July 2014

Some attempts to share pipelines

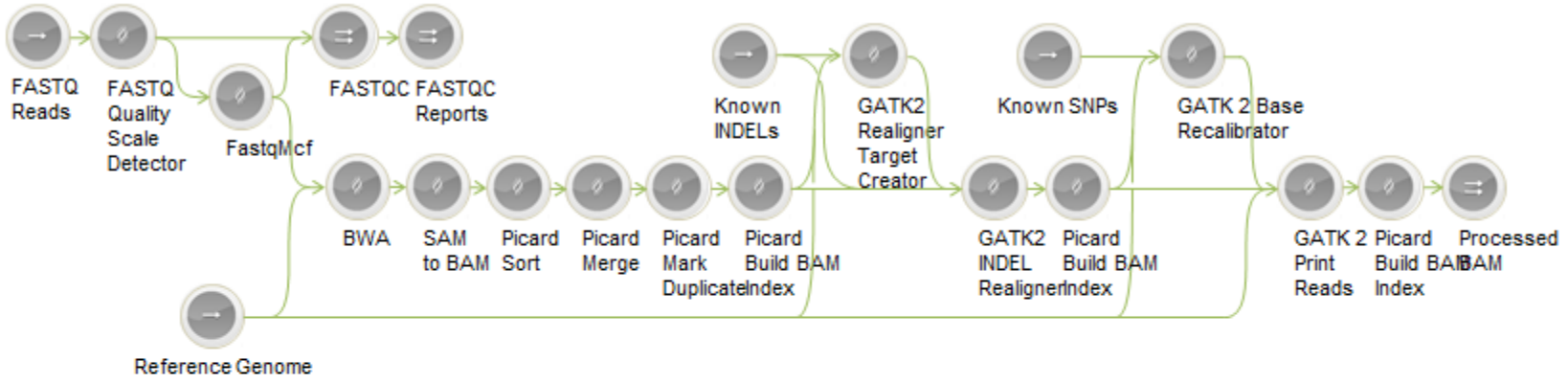
Paired end read alignment



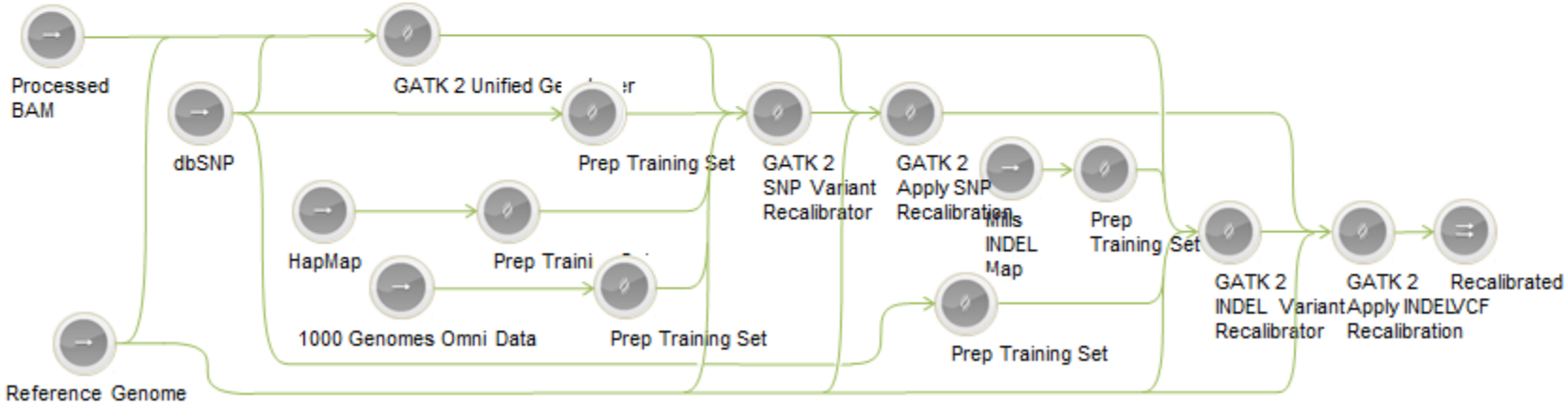
gkno.me

Variant calling

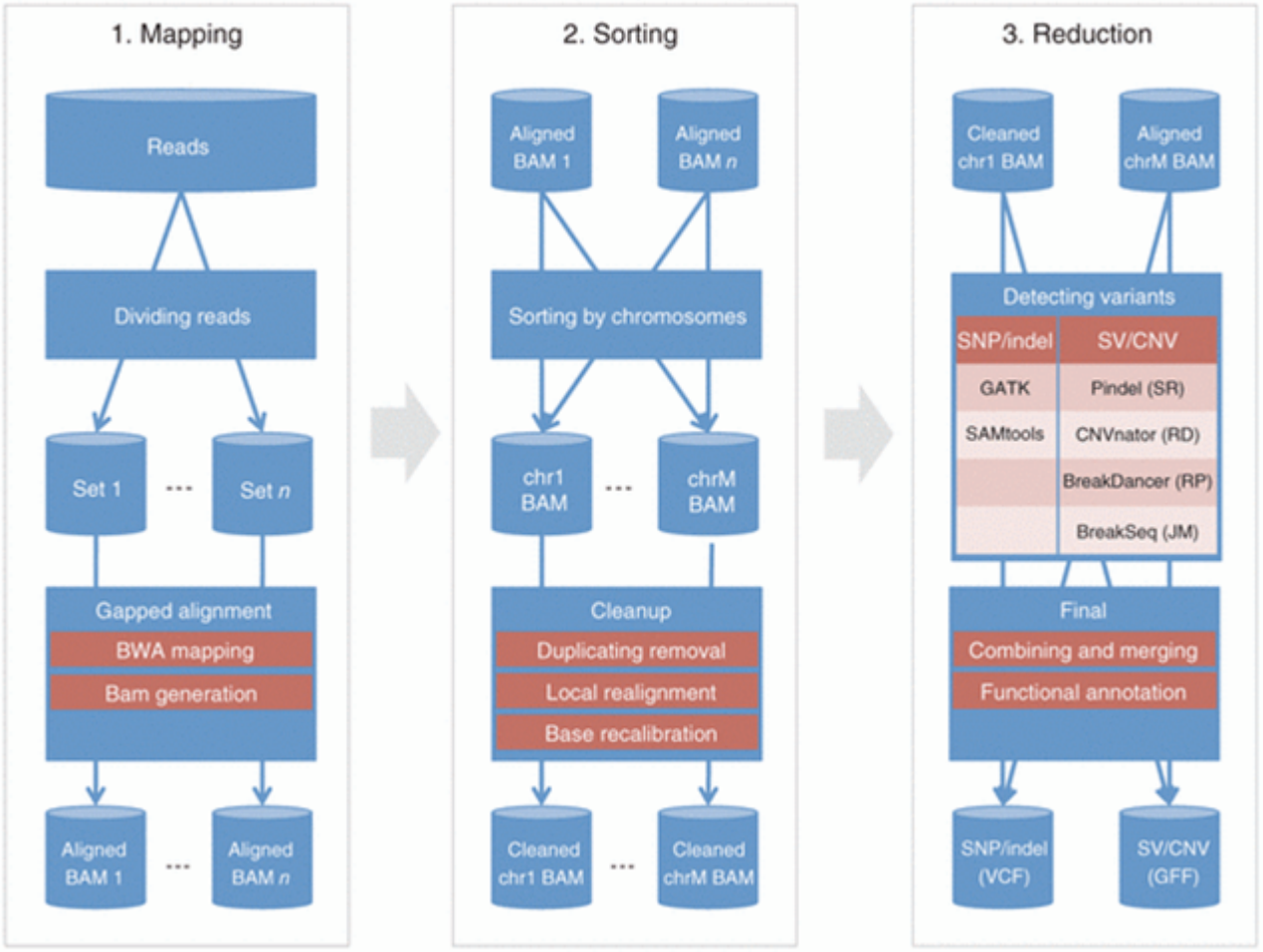
Part 1: Alignment and Base Quality Score Recalibration



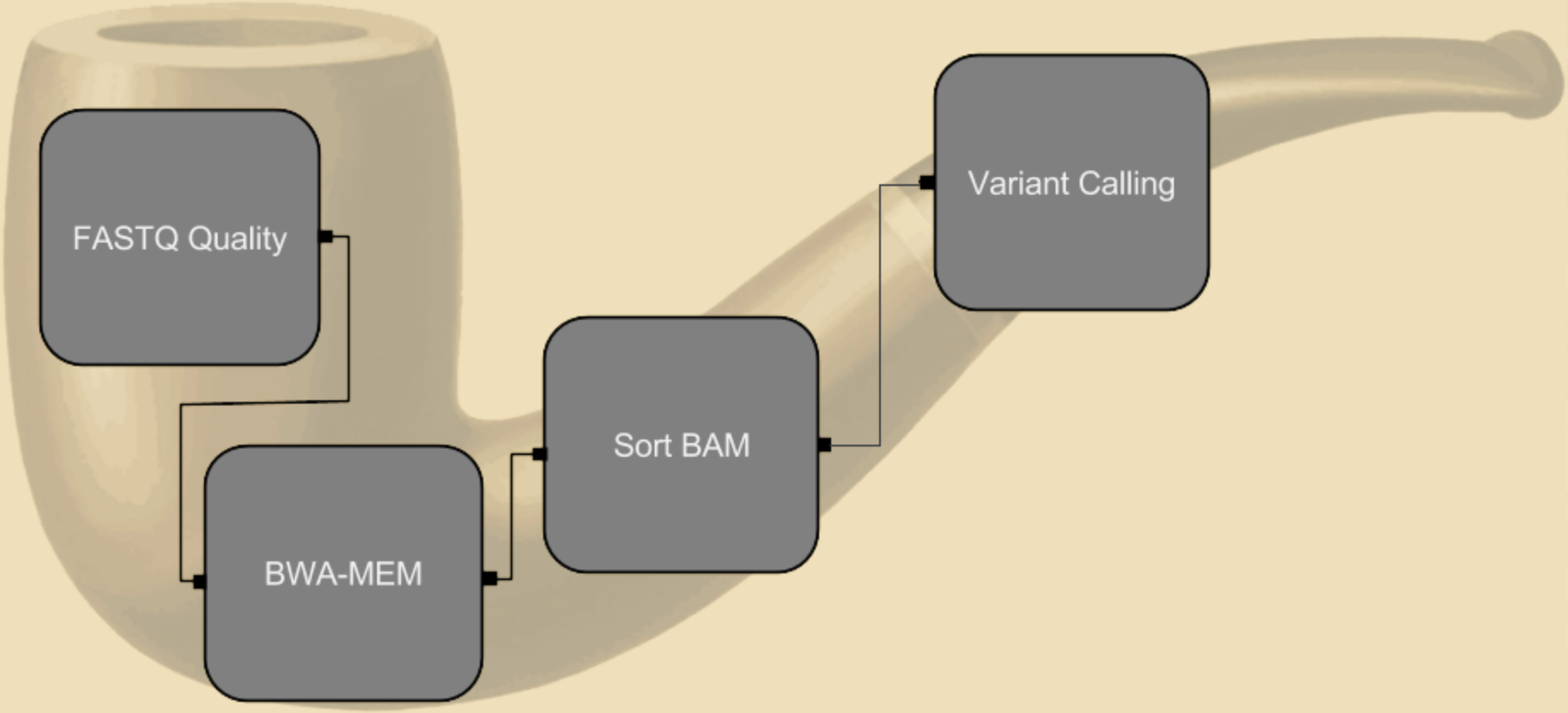
Part 2: Variant Calling and Variant Score Recalibration



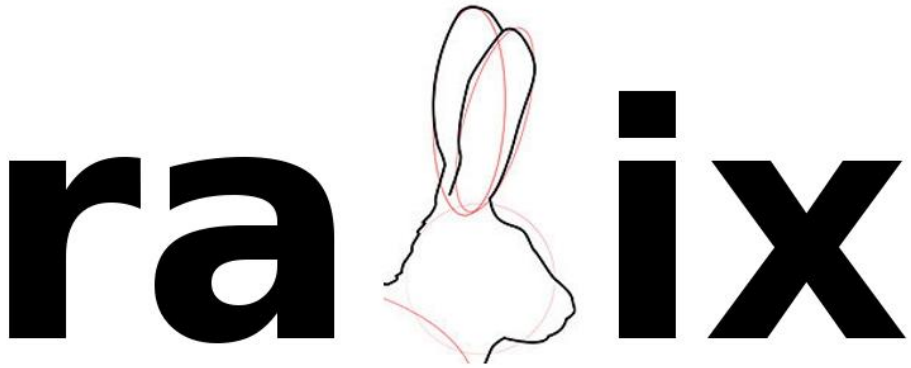
HugeSeq



http://hugeseq.hugolam.com/

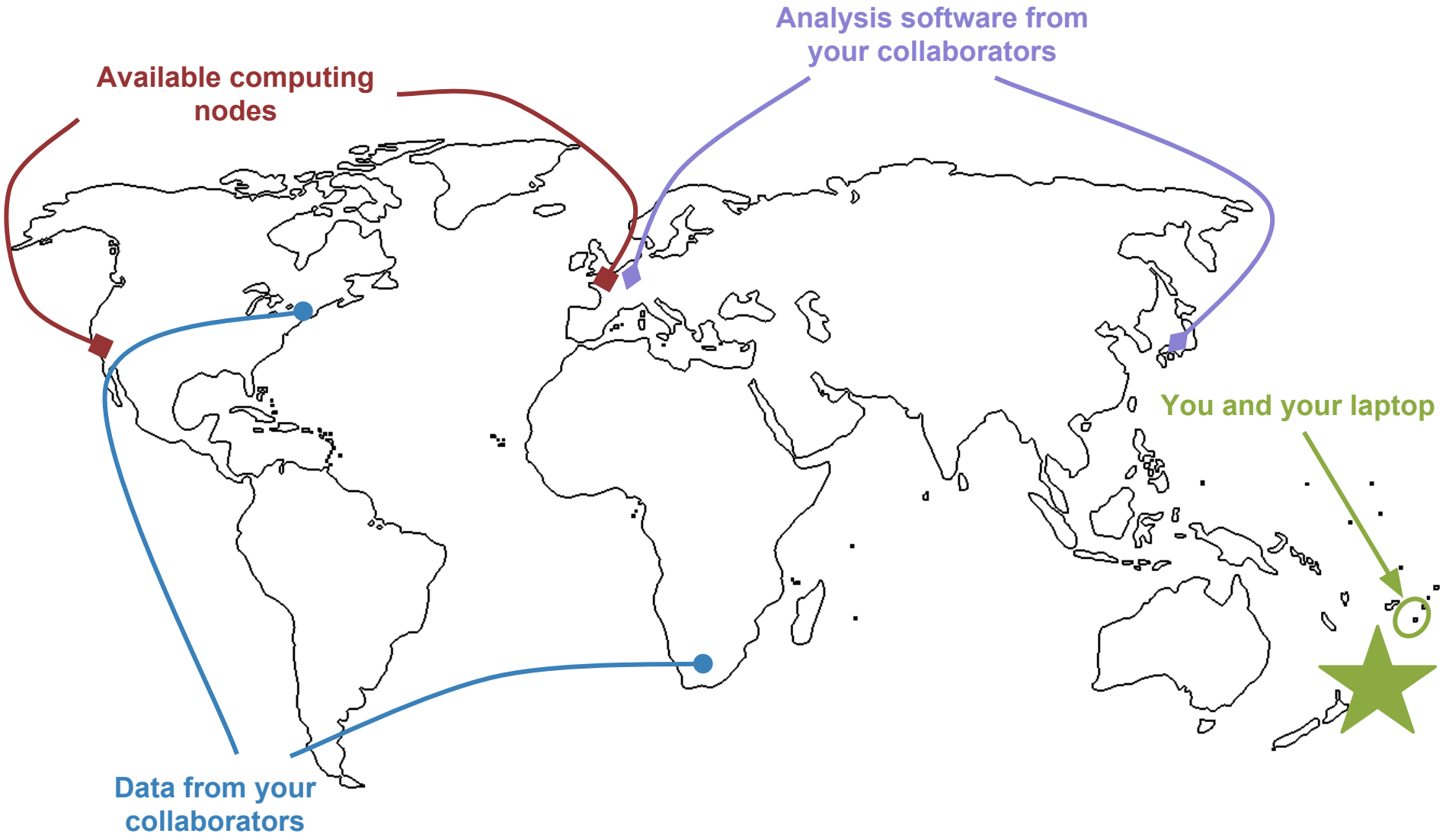


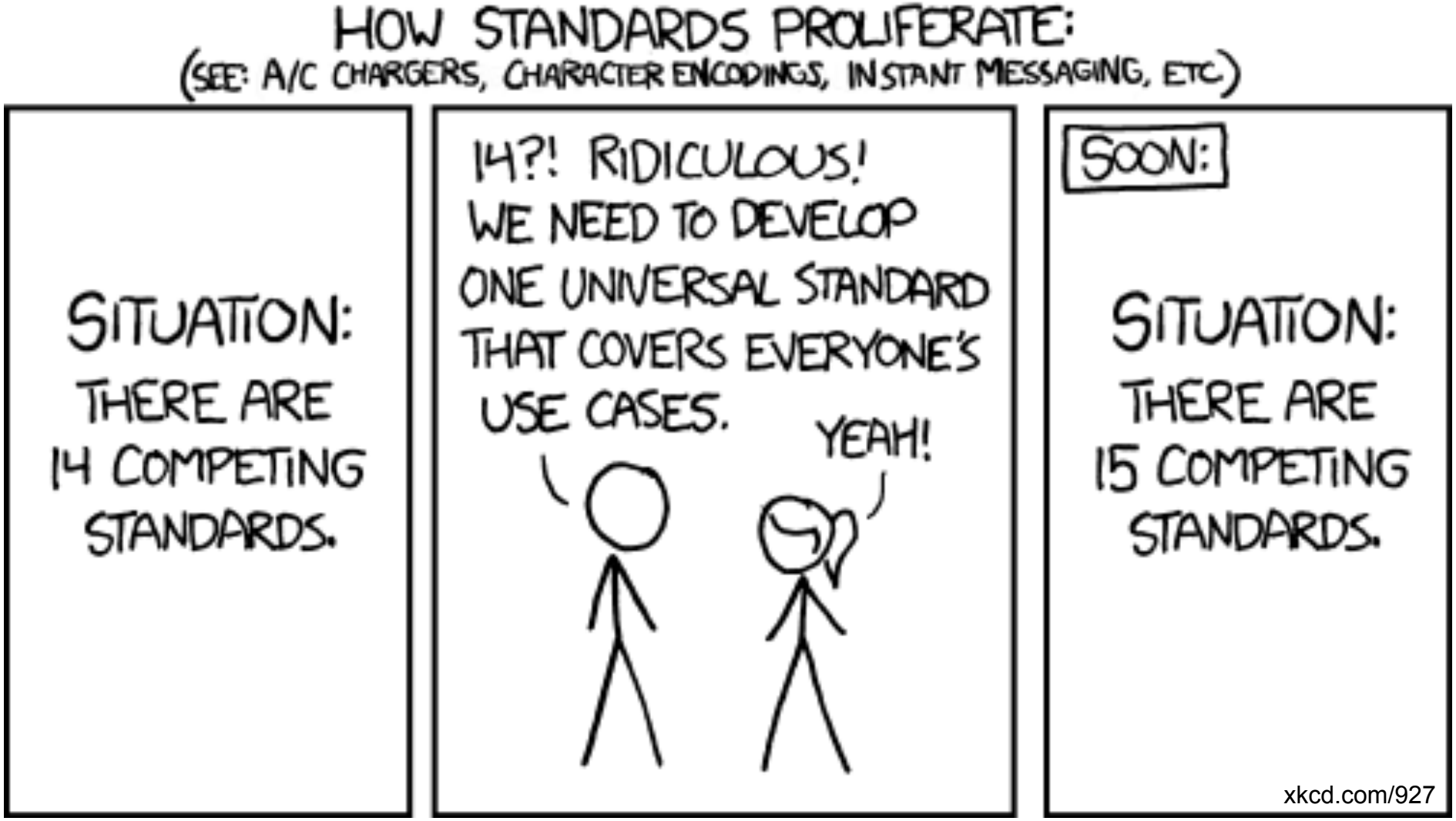
Ceci n'est pas une pipe.



- Reproducible Analyses for BioInformatics (eXperiments)
- Open initiative to enable easy sharing and efficient running of genomics tools and pipelines
- **rabix.org**

One command to bind them all




HOW STANDARDS PROLIFERATE:
(SEE: A/C CHARGERS, CHARACTER ENCODINGS, INSTANT MESSAGING, ETC)
SITUATION: THERE ARE 14 COMPETING STANDARDS.
14?! RIDICULOUS! WE NEED TO DEVELOP ONE UNIVERSAL STANDARD THAT COVERS EVERYONE'S USE CASES.
YEAH!
SOON:
SITUATION: THERE ARE 15 COMPETING STANDARDS.
xkcd.com/927

It's been tried?

Using Semantic Workflows to Disseminate Best Practices and Accelerate Discoveries in Multi-Omic Data Analysis

Gil et al. 2013

http://www.isi.edu/~gil/papers/gil-etal-hiai13.pdf

It's been tried?

Using Semantic Workflows to Disseminate Best Practices and Accelerate Discoveries in Multi-Omic Data Analysis

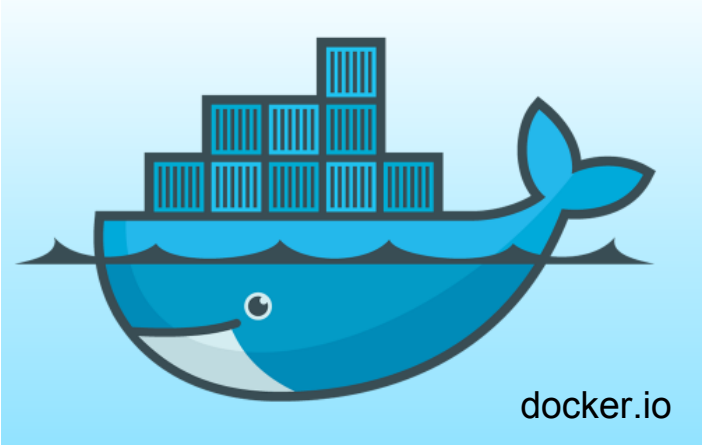
Gil et al. 2013

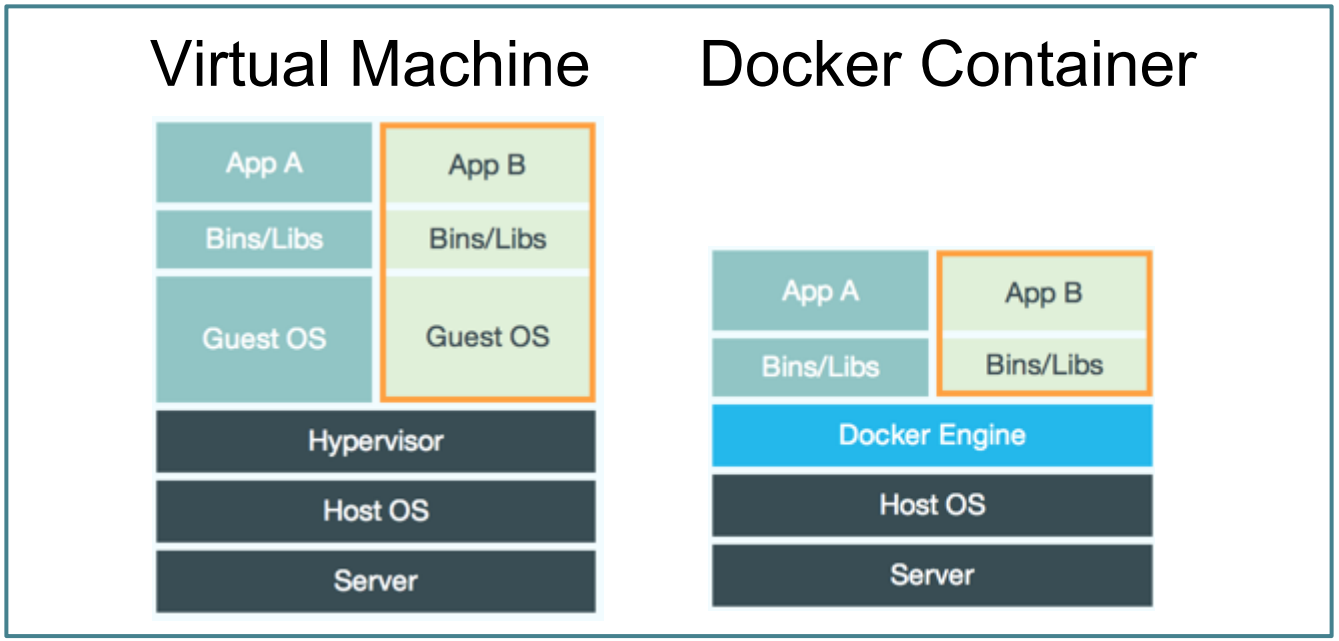
http://www.isi.edu/~gil/papers/gil-etal-hiai13.pdf

http://wings-workflows.org/

Docker

- Package tools in docker images
- Optionally, add adapter script in top layer





Describing tools

A well described tool

```
$ ./bwa mem
Usage: bwa mem [options] <idxbase> <in1.fq> [in2.fq]
Algorithm options:
       -t INT         number of threads [1]
       -k INT         minimum seed length [19]
       -w INT         band width for banded alignment
[100]
       -d INT         off-diagonal X-dropoff [100]
       [...]
```

BWA

- Software usage described in associated documentation materials and through CLI
- Default values are defined
- Parameter types are defined
- Follows POSIX guidelines
- Can be parsed!

Still not enough

```
$ ./bwa mem
Usage: bwa mem [options] <idxbase> <in1.fq> [in2.fq]
Algorithm options:
       -t INT         number of threads [1]
       -k INT         minimum seed length [19]
       -w INT         band width for banded alignment
[100]
       -d INT         off-diagonal X-dropoff [100]      BWA
       [...]
```

- Are positional arguments files?
- Result on stdout means streamable; what about inputs?
- In case of multiple outputs which files are results?
  - Example: aligned/unaligned
- How much RAM do I need?
- Is it single/multi threaded?
  - -t gives it away, but not automatic

Describing with documents

- JSON or YAML subset, please
- Inputs and outputs are documents
- Semantic schema for inputs/outputs
- Reference docker image
- Describe command line adapter
  - Transform inputs to process args
  - Create outputs document
  - Metadata?
  - ...or do it in a script!

Describing data

- Data passed as documents
- Leads to nice things:
  - Much more (semantic!) data than file headers
  - Constraints on input/output
  - Genealogy/provenance
  - Info on accompanying index files and similar
  - Easy file-specific args (e.g. M/F from "gender" key)
  - Can do "group by" / "for each" in pipelines
  - ...

- But really hard to get adoption

Declarative pipelines

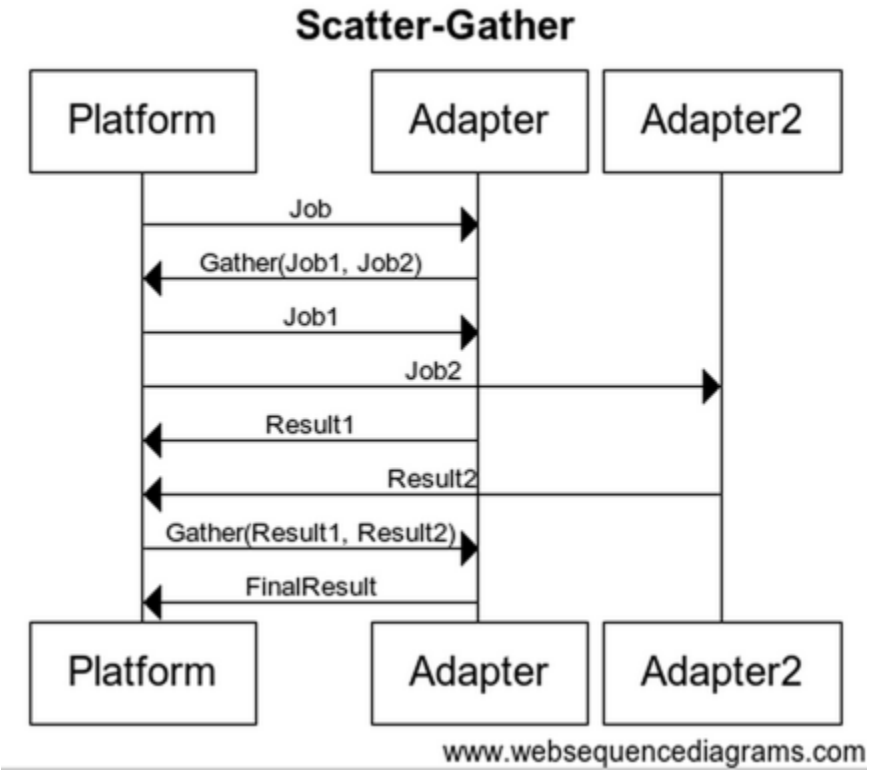
- Can use GUI for viewing/editing pipelines/runs
- Reference or embed other tools/pipelines as steps
  - Reference files or fragments, with checksum
- Can use pipeline input/output schema for interface abstraction
- Functionally pure, so can cache results

Clusters and clouds

- Parallelize the app (multi-node)
  - Scatter-gather
  - Existing capabilities (MPI, hadoop, ...)

- Discover and describe required resources
  - RAM, CPU, ports, disk space, ...

Jobs return jobs return jobs...



Solves more than scatter-gather

- Inspect inputs before deciding on needed resources
- Return replacement job if not enough resources (e.g. omkilled or no space)
- Delegate an "external resource job" to platform so you don't have to handle download yourself
  - And so that it can note you are using it!
- Job groups for MPI
  - Require open ports and shared storage for result

Resources and hints

- What's needed to get the job done?
  - RAM
  - CPU cores
  - Disk space
  - Open ports
  - Internet access

- Scheduler hints
  - Estimated time or throughput
  - Which inputs/outputs are streamable
  - Platform-specific hints

Reproducibility

Need complete run descriptions

*"But **even code release is insufficient** when the software is used to acquire new biological results for publication. Nearly all software has **user-defined parameters** that can or must be tuned to the data characteristics or analysis goals. These parameter values should be reported alongside the output. This practice helps ensure reproducibility even in the lab doing the work. Software developers can facilitate this process by providing macros to **record the software version and parameter settings** at the end of an analysis or by integrating such functionality into the software itself. The additional effort this logging requires is minimal compared to the benefits."*

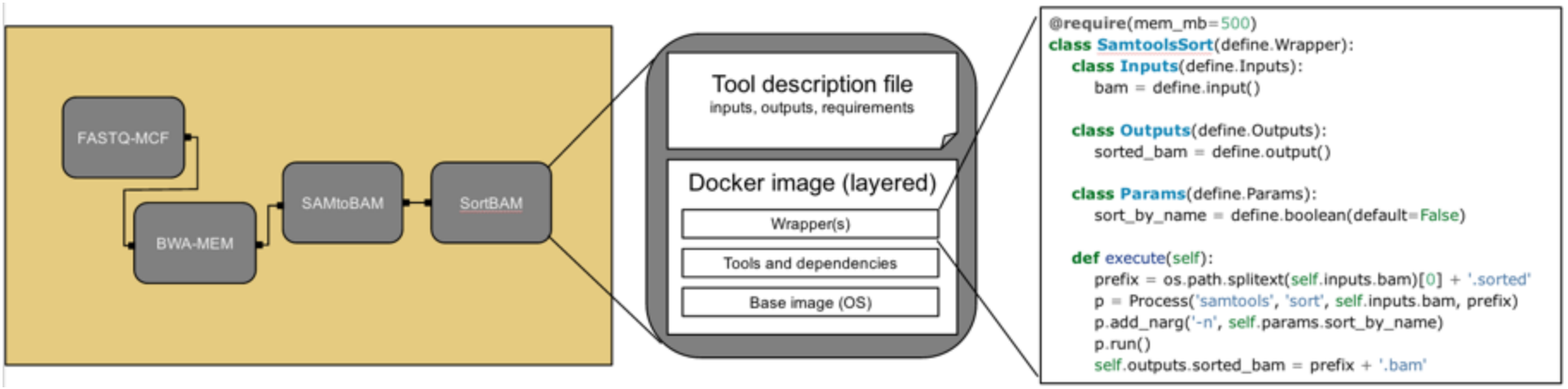
Nature Methods 2014

Easy with this approach

- Snapshots of all code and dependencies
- Image md5 is used as ID
- References to files/fragments have checksums
- Describe runs by referencing app and data
- Delegate fetching external (and possibly mutable) resources to platform so that the checksum can be stored and results optionally cached
- Simply share the files (or embed everything in one file)

Example

What it might look like



Installation

```
install:
  type: docker
  image_repo: http://rabix.org/bwa
  image_tag: 0.7.9
  image_id: md5hash
```

Inputs and outputs

```yaml
install:
  type: docker
  image_repo: http://rabix.org/bwa
  image_tag: 0.7.9
  image_id: md5hash
```

```yaml
inputs:
  type: object
  required: [reference, reads]
  properties:
    reference:
      type: dataset
    reads:
      type: array
      minItems: 1
      maxItems: 2
      items:
        type: dataset
    minimum_seed_length:
      type: integer
```

```yaml
outputs:
  type: object
  required: [sam]
  properties:
    sam:
      type: dataset
```

Wrapper

```
@require(mem_mb=5*1024, cpu=require.CPU_ALL)
class BWAMem(define.Wrapper):
    class Inputs(define.Inputs):
        reference = define.input(required=True)
        reads = define.input(required=True, list=True)

    class Params(define.Params):
        min_seed_len = define.integer()

    class Outputs(define.Outputs):
        sam = define.output()

    def _align(self, ref, reads, out_name):
        p = Process('bwa', 'mem', stdout=out_name)
        p.add_narg('-t', self.resources.cpu_count)
        p.add_narg('-k', self.params.min_seed_len)
        base_name = os.path.splitext(os.path.basename(ref))[0]
        p.add_arg(base_name)
        for read in reads:
            p.add_arg(read)
        p.run()

    def execute(self):
        self._align(self.inputs.reference, self.inputs.reads, 'result.sam')
        self.outputs.sam = 'result.sam'
```

Scatter-gather wrapper

```
class BWAMemSG(BWAMem):
    class Params(BWAMem.Params):
        reads_per_job = define.integer(default=100000, min=1)

    def _split_fq(self, path):
        prefix = os.path.splitext(os.path.basename(path))[0]
        Process('split', '-l', self.params.reads_per_job * 4, '--additional-suffix', '.fq', path, prefix).run()
        return sorted(glob.glob(prefix + '*.fq'))

    @require(cpu=1)
    def execute(self):
        chunks_1 = self._split_fq(self.inputs.reads[0])
        chunks_2 = self._split_fq(self.inputs.reads[1]) if len(self.inputs.reads) == 2 else []
        paired_chunks = itertools.izip_longest(chunks_1, chunks_2)
        jobs = [self.job('work', {'ref': self.inputs.reference, 'reads': pair}) for pair in paired_chunks]
        return self.job('merge', {'results': jobs})

    @require(mem_mb=5 * 1024, cpu=require.CPU_ALL)
    def work(self, ref, reads):
        self._align(ref, reads, 'chunk.sam')
        return 'chunks.sam'

    @require(cpu=1)
    def merge(self, results):
        Process('samtools', 'merge', 'result.sam', *results).run()
        self.outputs.sam = 'result.sam'
```

...or embedded adapter

```yaml
adapter:
  base_cmd: [bwa, mem]
  stdout: output.sam
  inputs:
    minimum_seed_length:
      order: 0
      type: named
      name: -k
    reference:
      type: positional
      order: 1
      transform: strip_ext
    reads:
      type: positional
      order: 2
  runtime:
    cores:
      order: 0
      type: named
      name: -t
  outputs:
    sam: output.sam
```

- Transform job input document to process description
- Create output document
- Use SPARQL rules?

Pipeline

```yaml
steps:
  index:
    app:
      $ref: bwa-index.yml                    # local reference
      checksum: sha1$hash
    inputs:
      fasta: fasta                           # pipeline input
  align:
    app:
      $ref: https://example.com/bwa-mem.yml  # remote reference
      checksum: sha1$hash
    inputs:
      reads: reads                           # pipeline input
      reference: index.indexed               # connect
    outputs:
      sam_file: sam                          # pipeline output
    parameters:
      skip_seeds: 1000
```

Run

```
$ rabix run https://s3.amazonaws.com/boysha/pipeline_bwa_freebayes.json
Usage: rabix run [-v] https://s3.amazonaws.com/boysha/pipeline_bwa_freebayes.json --read=<read_file>... --
reference=<reference_file>

Options:
  -v --verbose                          Log level set to DEBUG
  --read=<read_file>...                 Read sequence (or the first mate of a paired end read)
  --reference=<reference_file>          Reference sequence to which to align the reads

$ rabix run https://s3.amazonaws.com/boysha/pipeline_test_bwa_freebayes.json \
>   --reference https://s3.amazonaws.com/boysha/testfiles/example_human_reference.fasta \
>   --read https://s3.amazonaws.com/boysha/testfiles/example_human_Illumina.pe_1.fastq \
>   --read https://s3.amazonaws.com/boysha/testfiles/example_human_Illumina.pe_2.fastq
[...]
Output ID           File path                                                              File size
alignment           /home/boysha/wiwukojuvo.BwaMem.3/example_human_Illumina.bam            21402
vcf                 /home/boysha/wiwukojuvo.FreeBayes.1/example_human_Illumina.vcf          7990
```

Rabbits?

- Rabix won't be a platform
- Goal of the project is to keep all of this easy, accessible, portable and reproducible.

Run tools

- Run on localhost
- Run multi-node with shared storage

- Near-term:
  - Remove shared storage requirement
  - Run from the browser
  - Run on AWS
  - Run on Google Compute Engine

Development tools

- Python SDK
- CI/build server integrated with Github
- Open app registry

- Near-term:
  - Graphical pipeline viewer/editor
  - Automatic usage statistics

Join us on Github!

- Run some example pipelines
- Talk vocabularies/ontologies/features
- Open an issue to start discussion
- Login with Github on rabix.org to let us know you're interested
- github.com/rabix/rabix

Thanks



- Teammates
  - Sinisa Ivkovic
  - Milica Kadic
  - Luka Stojanovic
- Seven Bridges Genomics
- BOSC/Codefest

Links

- https://rabix.org
- https://github.com/rabix/rabix
- http://www.nature.com/nmeth/journal/v11/n3/full/nmeth.2880.html
- http://bio-bwa.sourceforge.net/
- http://gkno.me
- http://hugeseq.hugolam.com/
- http://www.renemagritte.org/the-treachery-of-images.jsp
- https://sbgenomics.com