BioJulia

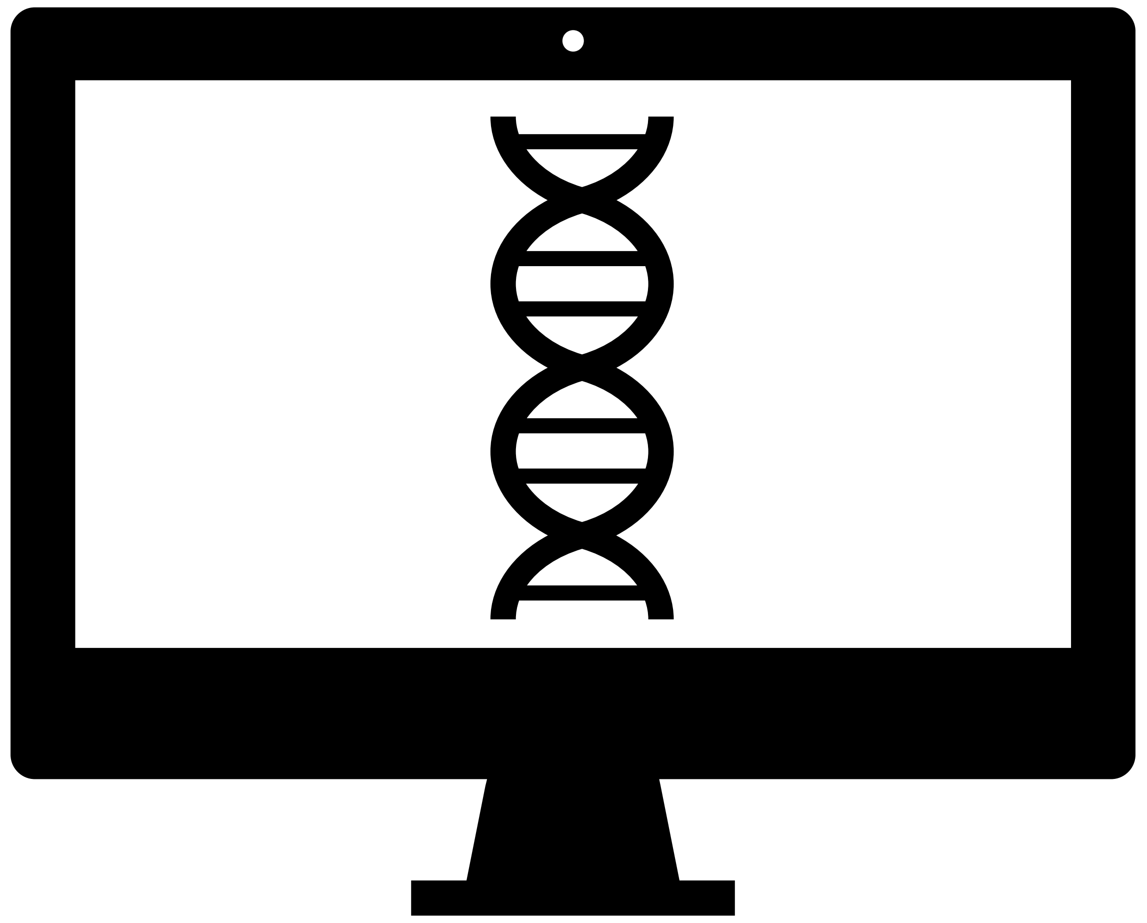
Design and scope

Jakob Nybo Nissen, 2021-10-21

Overview

- What do bioinformaticians do?
- What would the ideal bioinformatics framework look like?
- The values of BioJulia
- BioSequences

Bioinformatics as a discipline

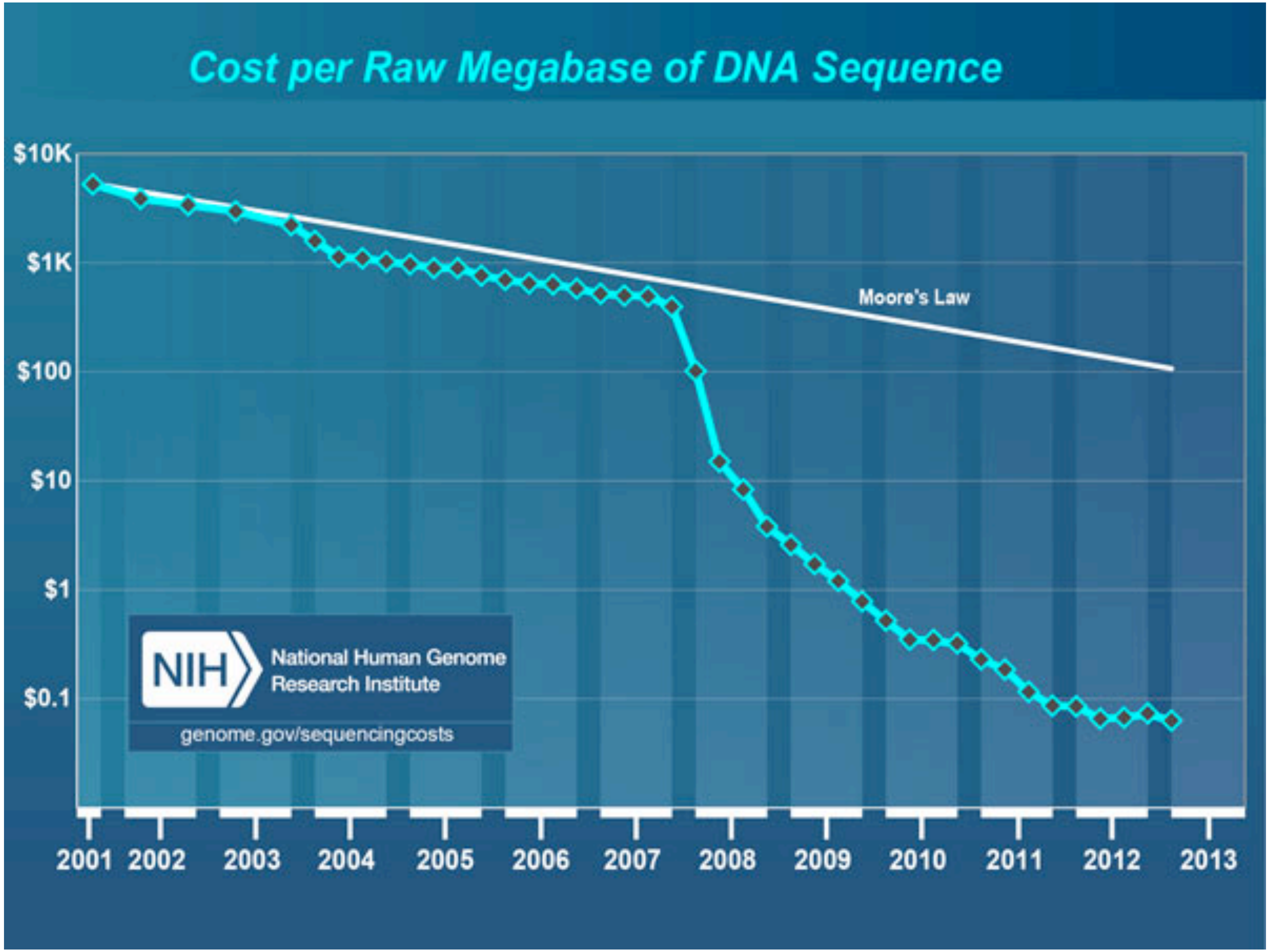


Field that develops methods and software tools for understanding biological data

Field that use and develop software for analysing biological sequences and molecules as digital data

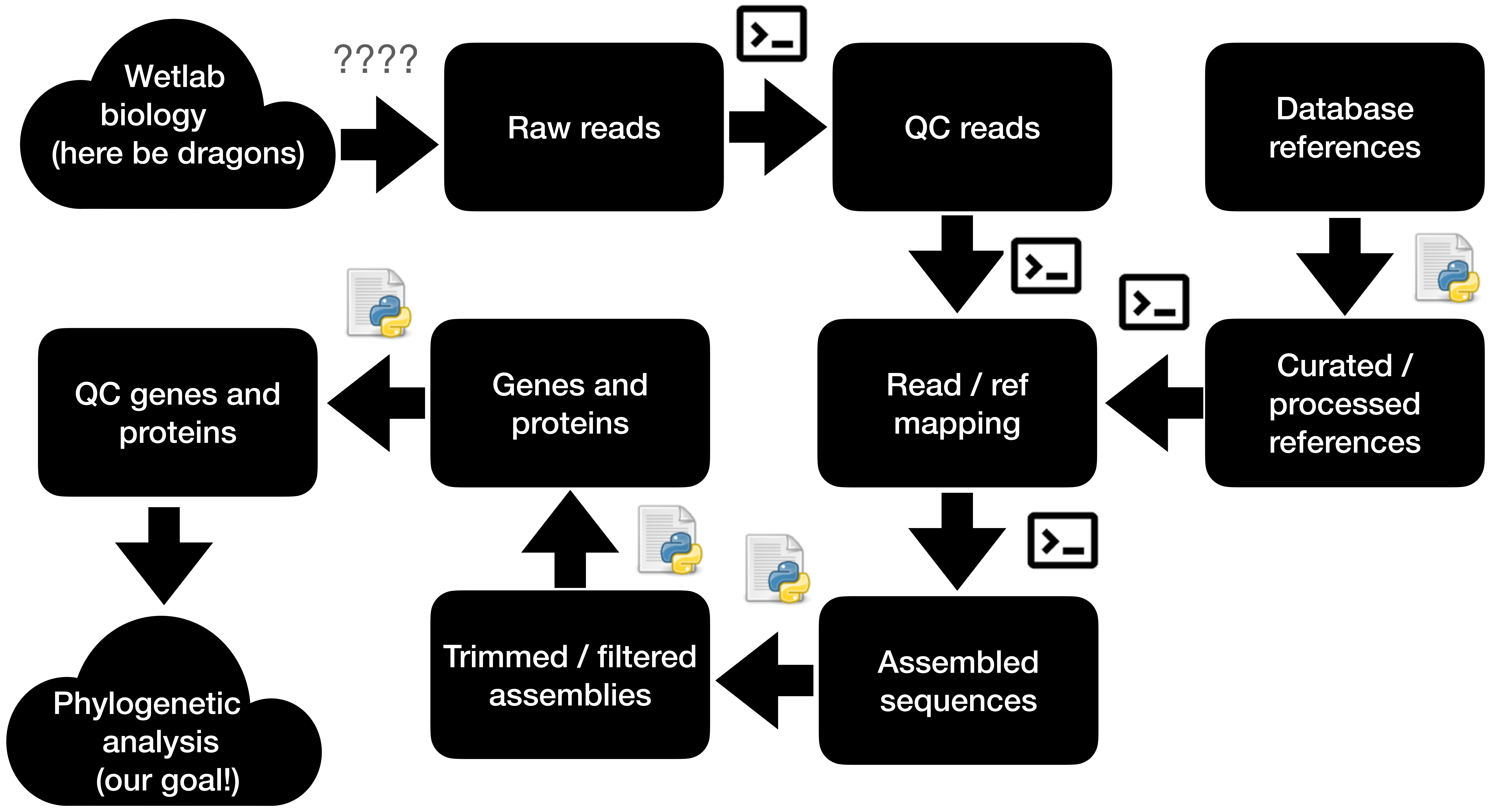
- Biological sequences *are* digital! No, really!
- This makes bioinformatics an "easy" field in some ways compared to most computational biology

The most abused graph in bioinformatics

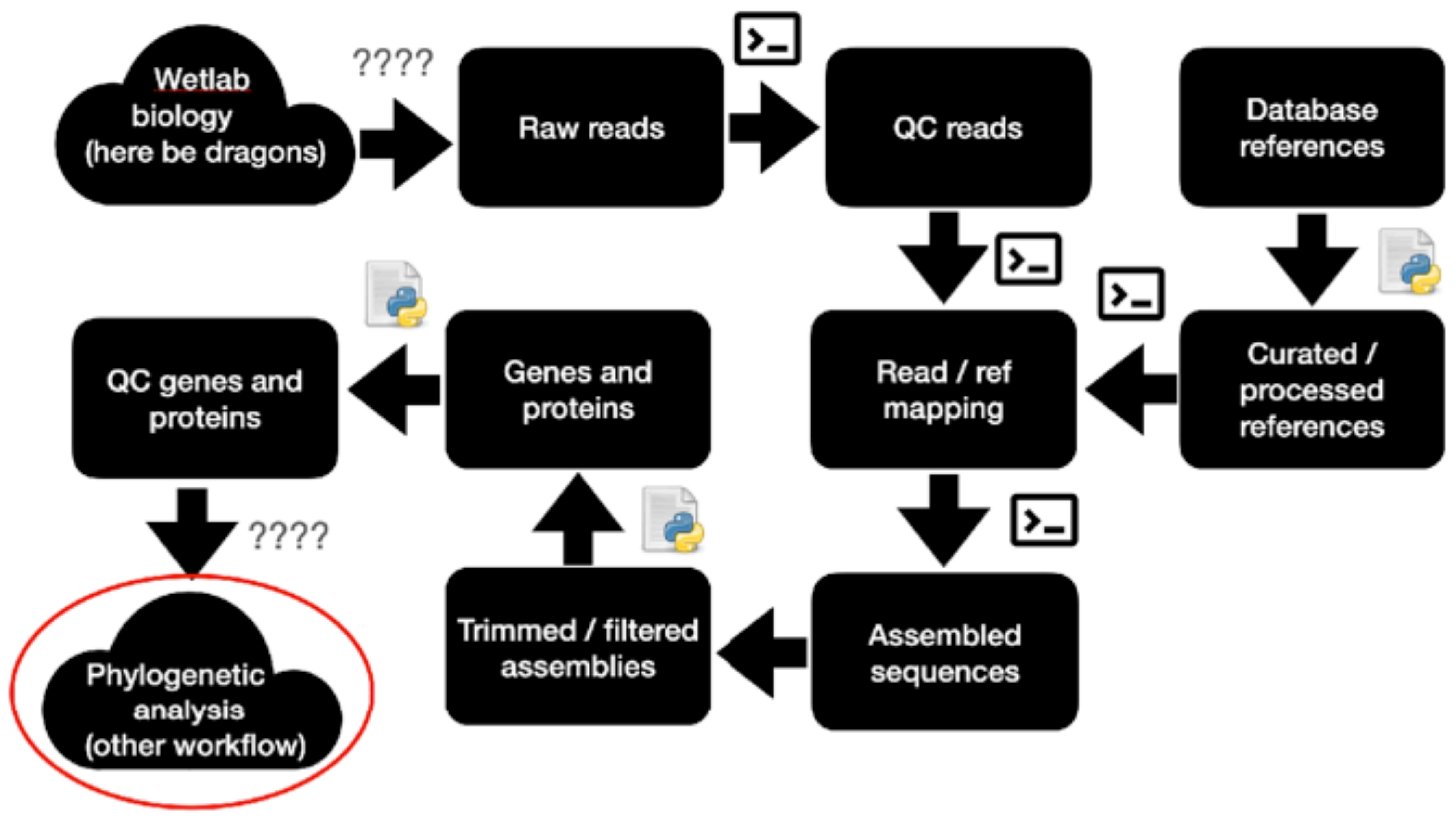


A similar story can be told about mass spectrometry, spatial data, and perhaps microfluidics?

Typical bioinformatics workflow

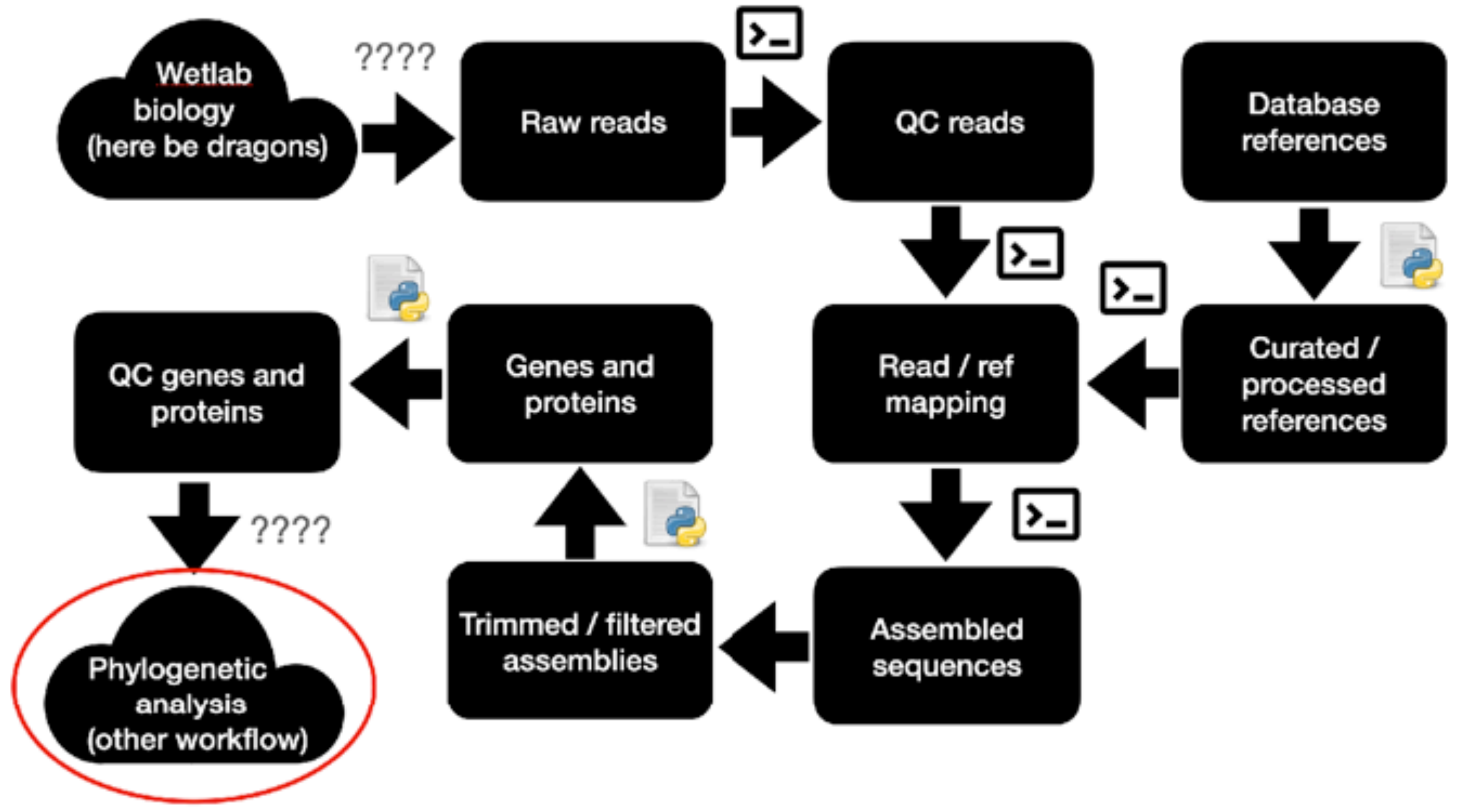

Wetlab biology (here be dragons)
????
Raw reads
QC reads
Database references
Curated / processed references
Read / ref mapping
Assembled sequences
Trimmed / filtered assemblies
Genes and proteins
QC genes and proteins
Phylogenetic analysis (our goal!)

Typical bioinformatics workflow

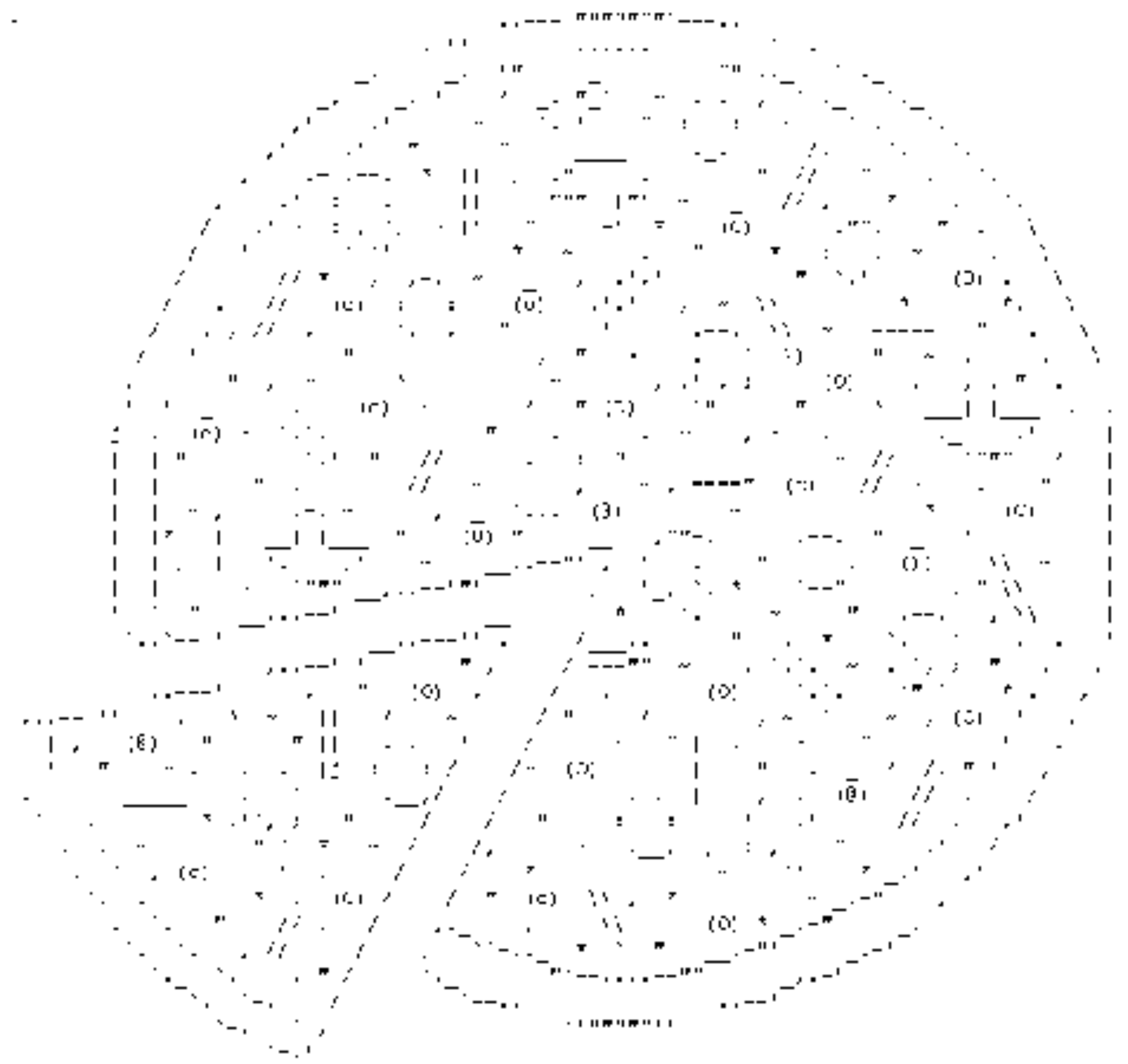

Wetlab biology (here be dragons)
????
Raw reads
QC reads
Database references
QC genes and proteins
Genes and proteins
Read / ref mapping
Curated / processed references
????
Phylogenetic analysis (other workflow)
Trimmed / filtered assemblies
Assembled sequences

- Workflows are mix of compiled tools and python/R scripts "duct taped" together
- Notoriously unreliable and brittle!
- Many tools *must* be in C/C++ because of the large amounts of data = speed!
- "Bag of tricks" approach all too common!
- But the large amounts of custom code and ever-changing requirements necessitates dynamic scripting languages

Typical bioinformatics workflow


Wetlab biology (here be dragons)
????
Raw reads
QC reads
Database references
Curated / processed references
Read / ref mapping
Genes and proteins
QC genes and proteins
????
Phylogenetic analysis (other workflow)
Trimmed / filtered assemblies
Assembled sequences

- Each step reads in data from disk, processes, and serialises back
- This leads to a *huge* amount of different file formats, and parsing problems
- Formats are often underspecified and ad-hoc
    - Perhaps deliberately?



Valid DNA sequence file, according to commonly used software

What would a solution look like?

- We need a *fast, dynamic* scripting language
- Rewrite ~~it~~ In Julia?
  - Completely unrealistic. Decades of tools, too much inherent knowledge
  - I can't even rewrite my own PhD project, realistically!
- Instead, improve the *glue language* aspect:
  - Fast enough to never need to use another language due to speed
  - Good at calling into shell and interface with other languages
  - Excellent at parsing
- We also need good "workflow management software", but that's another topic....

BioJulia

- Fast enough to not need C for speed
- Dynamic enough to not need Python for scripting
- Implement *basic, commonly used* bioinformatics algorithms
    - These are the "center of the wheel" of many analyses!
- Types and algorithms should be generic enough to be re-usable across a wide range of applications and tools: Don't re-invent the wheel too many times
- Have great file readers/writers
- Be the swiss army knife of bioinformatics
- Incremental replacement of Python/C with Julia.

BioSequences.jl

- Biological sequences *define* bioinformatics. Very important package!

bioseq is a `str`!"

`ACG"`

oseq is a `&[u8]`!"

`[` `65,67,71]`

- A biological sequence is a biological sequence!
- Internal representation is incidental. Abstract it away unless speed is necessary!
- "Parse, don't validate"

BioSequences.jl

```
BioSequence{Alphabet}
```

- `BioSequence`: Abstract type that all concrete biosequence types subtype
- This abstracts over its representation.

- `Alphabet`: Abstract type all alphabets subtype.
- An alphabet is the set of biological symbols that can be contained in the sequence
- Abstracts over "validation/parsing".
- The `Alphabet` + `BioSequence` type decide how data is encoded concretely.

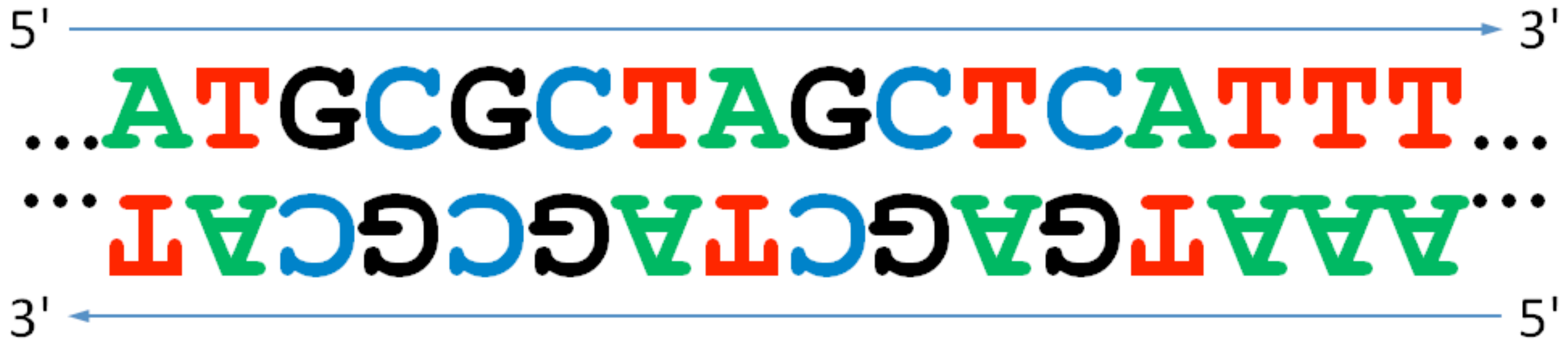
BioSequences.jl

`LongSequence{DNAAlphabet{4}}`

- `LongSequence`: Sequence is arbitrary length, mutable, stored on the heap in a Vector. Elements are encoded by bit-packing `UInt64`.
- You will use another kind of `BioSequence` next exercise!
- `DNAAlphabet{4}`: Sequence can contain `"-ACMGRSVTWYHKDBN"`. Each symbol is packed into 4 bits.
- Other pre-defined alphabets include `RNAAlphabet{2}` (`"ACGU"`) and `AminoAcidAlphabet` (`"ARNDCQEGHILKMFPSTWYVOUBJZX*-"`)

Things to do with biological sequences

Reverse complementation



Transcription and translation

```
DNA: TAG CTA GGC TAA ACA TAG

RNA: UGG CUA GGC UAC UCU UAG

AA :  W   L   G   Y   S  STOP
```

Questions?

Exercise 2