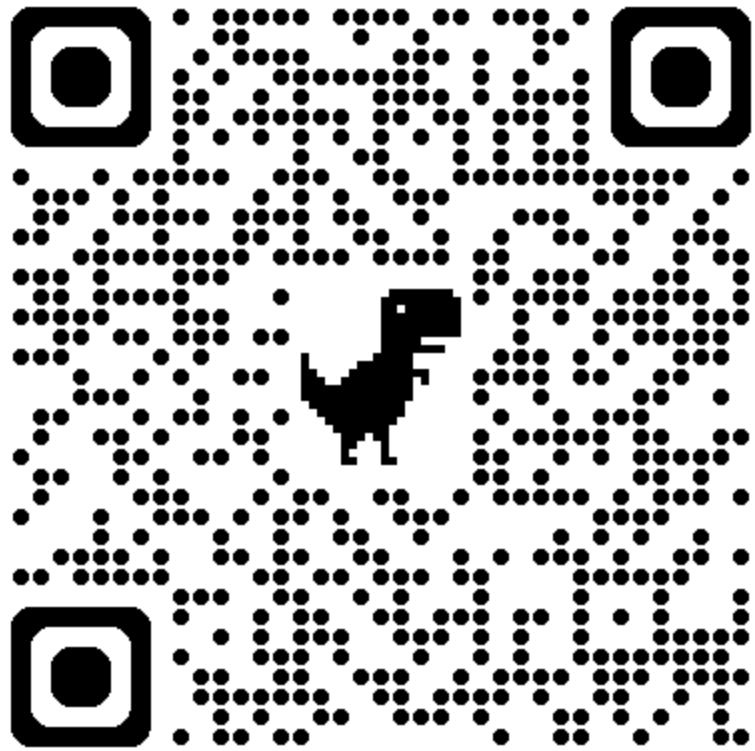


# Client-side bioinformatics: building robust bioinformatics tools for the browser with WebAssembly

ABACBS 2025, Adelaide, 27-11-25

<https://wasmmodic.github.io/>



ABACBS 2025

## Client-side Bioinformatics

*Building bioinformatics tools for the browser with WebAssembly*

```
~ > csvtk head iris.csv | csvtk pretty
sepal.length  sepal.width  petal.length  petal.width  variety
-----  -----  -----  -----  -----
5.1      3.5       1.4       .2        Setosa
4.9      3          1.4       .2        Setosa
4.7      3.2       1.3       .2        Setosa
4.6      3.1       1.5       .2        Setosa
5         3.6       1.4       .2        Setosa
5.4      3.9       1.7       .4        Setosa
4.6      3.4       1.4       .3        Setosa
5         3.4       1.5       .2        Setosa
4.4      2.9       1.4       .2        Setosa
4.9      3.1       1.5       .1        Setosa
~ >
```

Modified from [sandbox.bio](#)

ABACBS Wasm Workshop

# What is WebAssembly?

---

Binary instruction format for a stack-based virtual machine.

Runs at near-native speed inside browsers and outside (Node.js, WASI).

Language-agnostic: compile from C, C++, Rust, Go.



# Who are we?



Wytamma Wirth



Andrew Lonsdale



Leo Featherstone



Torsten Seemann

- Bioinformaticians with an interest in making tools and methods Accessibility
- Not WebAssembly experts but enthusiasts

# Workshop Overview

Section	Presenter	Time
Introduction to WebAssembly	Wytamma Wirth	1:30 - 1:40 PM
History of WebAssembly	Torsten Seemann	1:40 - 1:55 PM
Intro to Web Development	Leo Featherstone	1:55 - 2:05 PM
Bioinformatics Use Cases for WebAssembly	Wytamma Wirth	2:05 - 2:25 PM
Practical Walkthrough: Compiling Rust to Biowasm	Andrew Lonsdale	2:25 - 2:40 PM
Demos, Hackathon Explanation, and Setup	Wytamma Wirth	2:40 - 3:00 PM
Break		3:00 - 3:30 PM
Hackathon Time	All Participants	3:30 - 5:00 PM

# Motivation



Make bioinformatics more accessible



Increase the reach and impact of your tools

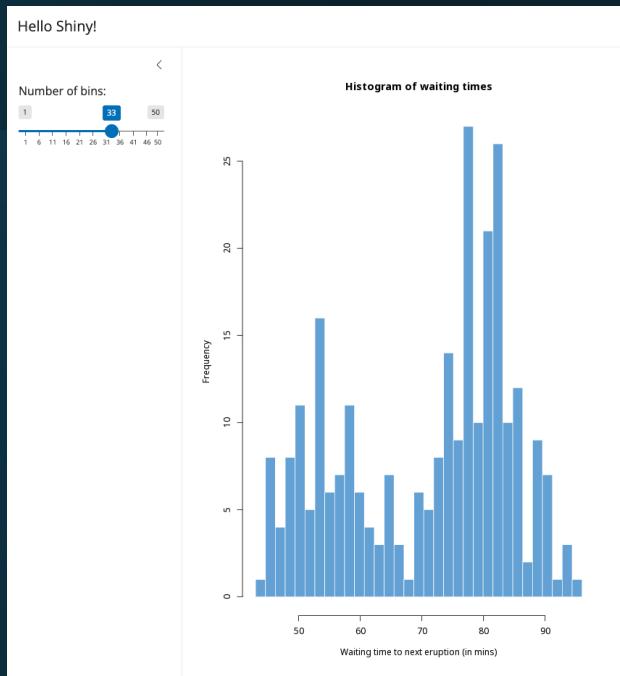


Improve reproducible science



Reduce code duplication

# You will have a URL that you can send to your Mum/Boss at the end of this Workshop



Interactive R code editor

Run code

```
1 fit <- lm(mpg ~ am, data=mtcars)
2 summary(fit)
```

Call:

```
lm(formula = mpg ~ am, data = mtcars)
```

Residuals:

Min	1Q	Median	3Q	Max
-9.3923	-3.0923	-0.2974	3.2439	9.5077

Coefficients:

	Estimate	Std. Error	t value	Pr(> t )
(Intercept)	17.147	1.125	15.247	1.13e-15 ***
am	7.245	1.764	4.106	0.000285 ***

---

Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.'

Residual standard error: 4.902 on 30 degrees of freedom  
Multiple R-squared: 0.3598, Adjusted R-squared:  
F-statistic: 16.86 on 1 and 30 DF, p-value: 0.00028

**Muscle WebAssembly**

MUSCLE (MUltiple Sequence Comparison by Log-Expectation) running entirely in your browser via WebAssembly. This client-side implementation provides fast, accurate multiple sequence alignment without requiring server uploads - your data never leaves your computer.

**Input Sequence**

```
>test1
MGDVEKGKKIFIMKCSQCHTVEGGKHKTGPNLHGLFGRKTGQAPGY
SYTAANKNKGIIW
GEDTLMEYLENPKKYIPGTMIFVGIKKEERADLIAYLKKATNE
```

Upload File

Use the example Clear sequence

**Output Format**

Clustal

Run Alignment

Open source on GitHub

# Who are you?

Join at [menti.com](https://menti.com) | use code 5625 1623

 Mentimeter

Go to

**www.menti.com**

Enter the code

**5625 1623**



Or use QR code

What is  
Web Assembly ?

# Torsten Seemann



THE UNIVERSITY OF  
MELBOURNE



Doherty  
Institute



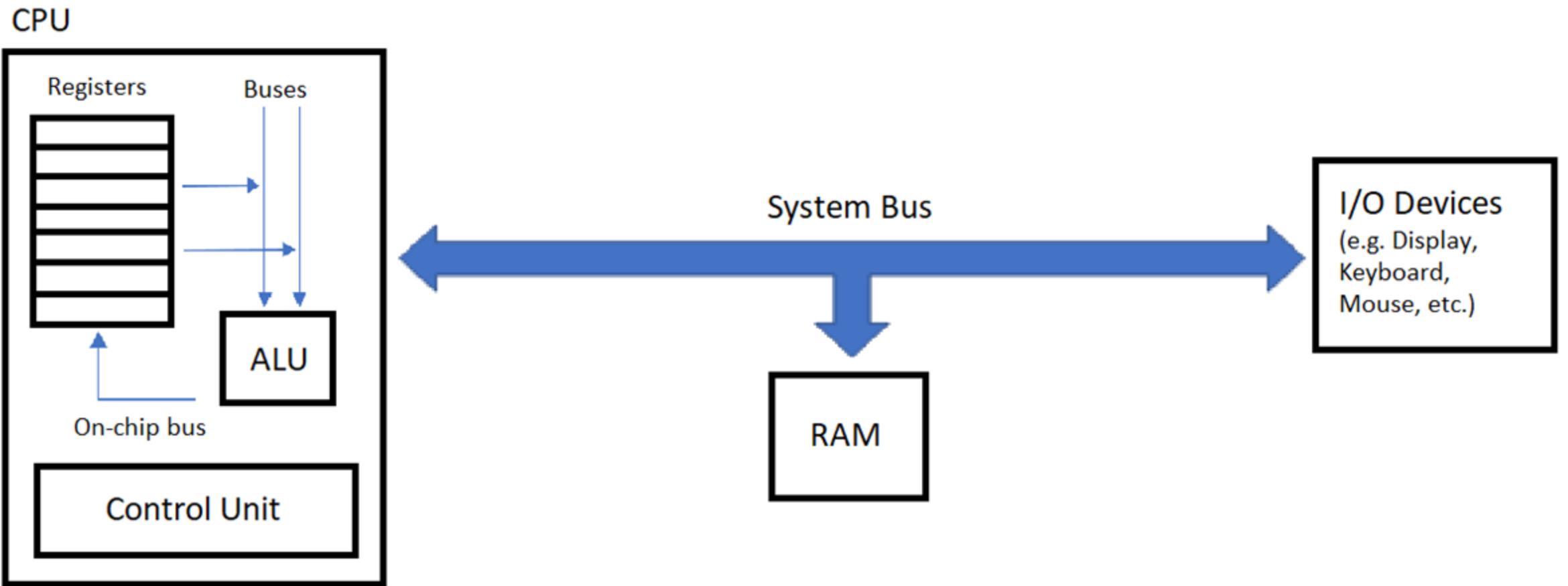
CENTRE FOR  
PATHOGEN  
GENOMICS



# What is assembly?



# Computer architecture



## CPU architecture

- Famous 8-bit CPU - MOS 6502
- Only 3 registers (8-bit) - A, X, Y
- Address 64K RAM (16-bit)
- 56 core instructions, 151 total



## CPU opcodes

```
A9 2A      # Put 42 in the “A” register  
18          # Clear the carry flag  
69 0F      # Add 15 to the “A” register  
8D 00 C0    # Store “A” in RAM at 49152
```

## Opcode mnemonics for humans

LDA #42        # Put 42 in the “A” register

CLC              # Clear the carry flag

ADC #15        # Add 15 to the “A” register

STA 49152      # Store “A” in RAM at 49152

Compile .asm to machine code

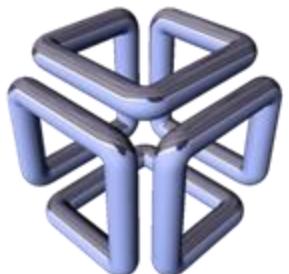
```
.byte power 100      # variable in RAM
loop:                # label (pointer)
    LDA power        # load var into “A”
    SBC 2            # subtract 2 from A”
    STA power        # store back in var
    BNE loop         # loop back “IF” not zero
    RTS              # return to caller
.end                # stop assembling
```

Write once, run anywhere

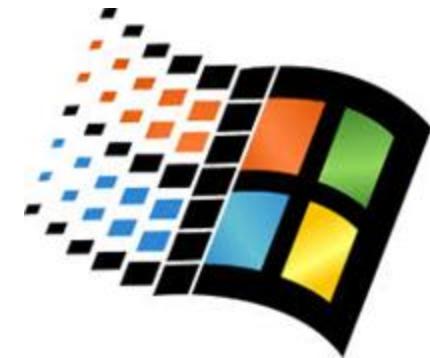
Java - the future in the 1990s



## *Java Runtime Environemtn (JRE)*



digital

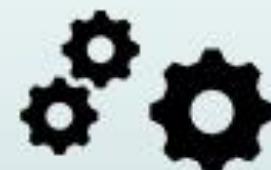




# Java Virtual Machine



Java Source File



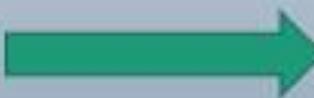
Java Compiler



Java Class File

Java Virtual Machine

Interpreter



Operating System



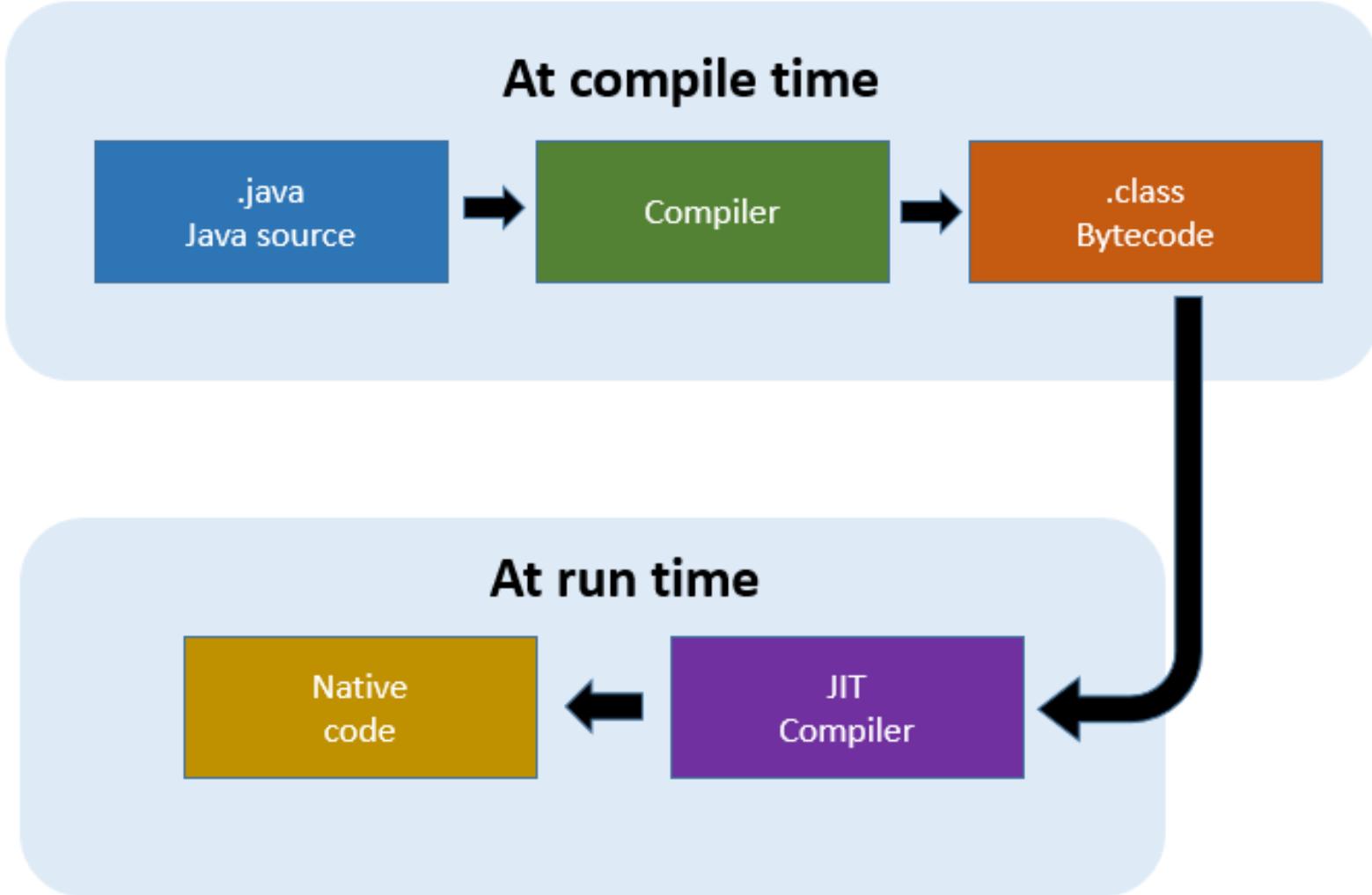
## Java byte code ~ CPU opcodes

```
// Bytecode stream: 03 3b 84 00 01 1a 05  
68 3b a7 ff f9  
// Disassembly:  
  
iconst_0           // 03  
istore_0          // 3b  
iinc 0, 1         // 84 00 01  
iload_0           // 1a  
iconst_2          // 05  
imul              // 68  
istore_0          // 3b  
goto -7           // a7 ff f9
```

### JVM

Runs a virtual CPU and  
Interprets bytecodes at runtime

## Just In Time Compilation (JIT)



# JavaScript - no bytecode, interpreted



*JS Interpreter*



# WASM (Web Assembly)

Cross-platform - works on all

# BIOCONDA®

*UNIX Compatibility Layer - but compiled for a specific CPU*



Cross-platform - works on all



**WEBASSEMBLY**  
FAST, SECURE, AND  
PORTABLE

***WASM Compatibility Layer - independent of OS or CPU***



## WASM opcodes

- `0x00 : unreachable` - Indicates a point in the code that should not be reached.
- `0x02 : block` - Defines a structured control block.
- `0x41 : i32.const` - Pushes a 32-bit integer constant onto the stack.
- `0x6a : i32.add` - Adds two 32-bit integer values from the stack.
- `0x28 : i32.load` - Loads a 32-bit integer from memory.

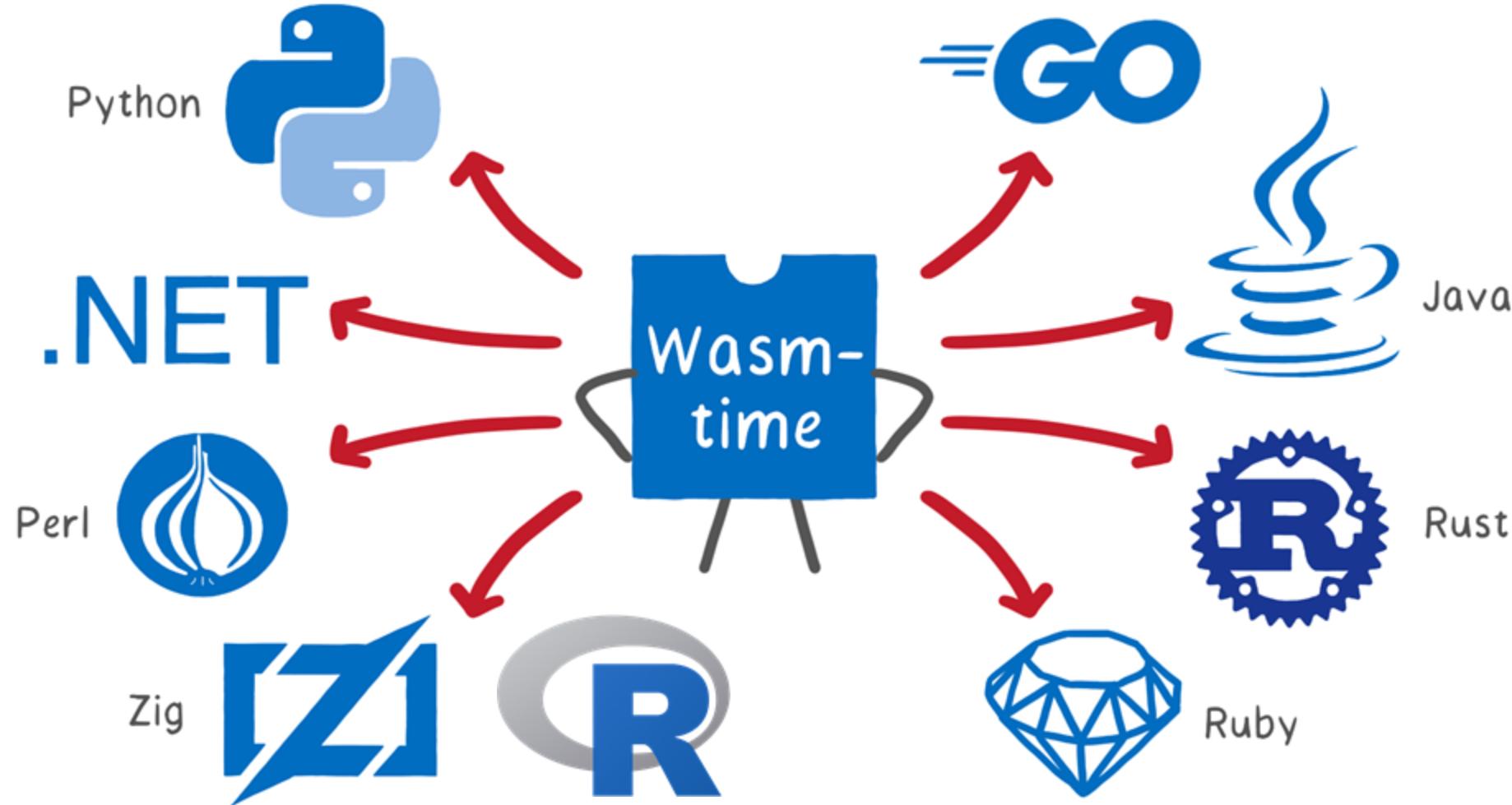
- “Stack based” architecture, strong typing
- Maps well to our register-based CPUs
- Supports use of SIMD and GPU

## WASM assembly language

```
i32.const 42          # push 42 onto stack
i32.const 15          # push 15 onto stack
i32.add              # pops 42 and 15, pushes sum

loop                 # start a loop
...
    br_if 0          # break if top of stack is zero
end                 # end loop
...                  # rest of program
```

WASM = the new common CPU ?



Thank you

## **For those new to webdev**

*... and why you might  
(should) choose WASM*

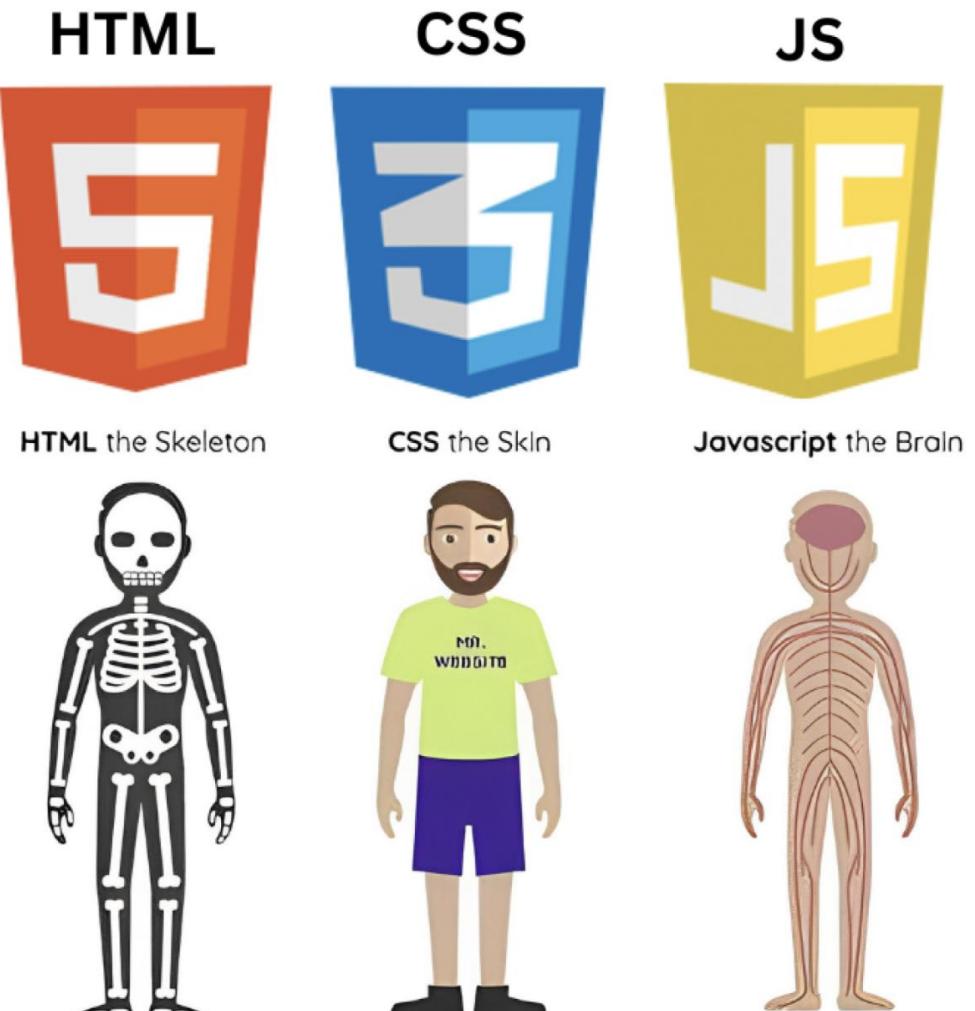
- JavaScript is the language of the web
- Yes, a language famously written in 12 days holds the world together

## **The Birth & Death of JavaScript**

A talk by Gary Bernhardt from PyCon 2014



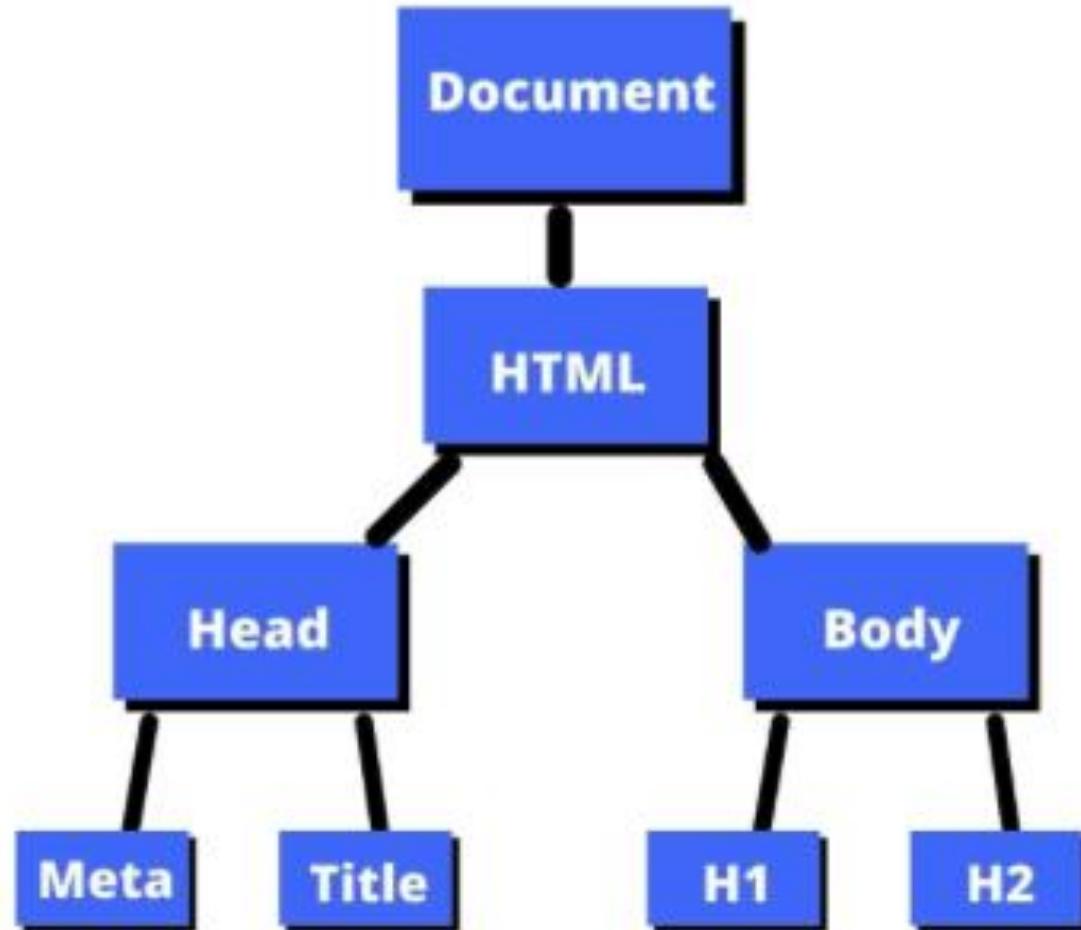
# JavaScript – CSS – HTML



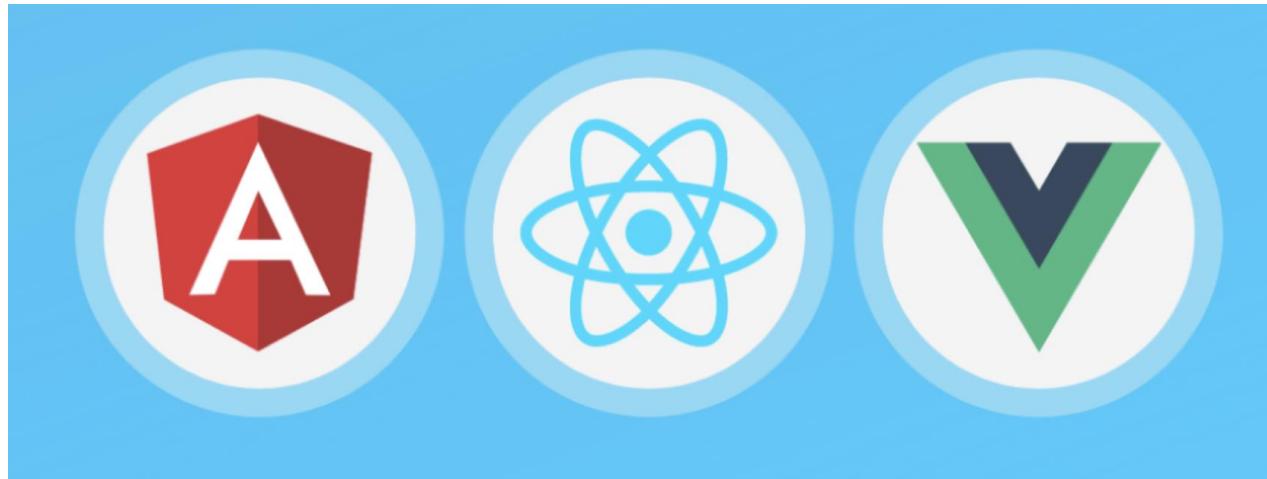
*An analogy between HTML, CSS, and JavaScript*

# **The Document Object Model (DOM) connects everything**

- DOM = tree representation of the document reflecting the way HTML elements are nested
- Browser navigates the DOM to apply JS/WASM code
- CSS applied to different elements in HTML structure
- HTML 'class/id' tag are nodes!



# Frameworks help abstract this away



These frameworks are in JS, so ultimately are syntactic sugar.  
They basically modify the DOM

# WASM replaces the functional JavaScript

- Via "bindings" to JS



# So why use WASM in place of JS?

- Javascript is **NOT** a good language for scientific programming
  - No integer type
  - Few standard libraries good for science
  - Parsing data is a pain
  - Not statically typed (But you can use typescript)
    - But typescript offers only the development benefits of typing, not the speed benefits

AND more access to full hardware, not just JS interpreter/runtime. Turns your browser into computers with near-native speed

# An example: if I had my time again



**PhyloJS**

Release passing downloads 2k npm v1.12.0 issues 6 open codecov 89% semantic-release commitizen friendly

PhyloJS is a powerful javascript/typescript library for manipulating phylogenetic trees. It allows users to read and write trees in various formats, extract subtrees, and compute properties such as the most recent common ancestor (MRCA) of a set of nodes, among other features.

PhyloJS is especially helpful in situations where you need to analyze large phylogenetic trees, such as when you're studying evolutionary relationships in bioinformatics or computational biology. PhyloJS is lightweight and has zero dependencies.

### Installation

You can install PhyloJS using npm:

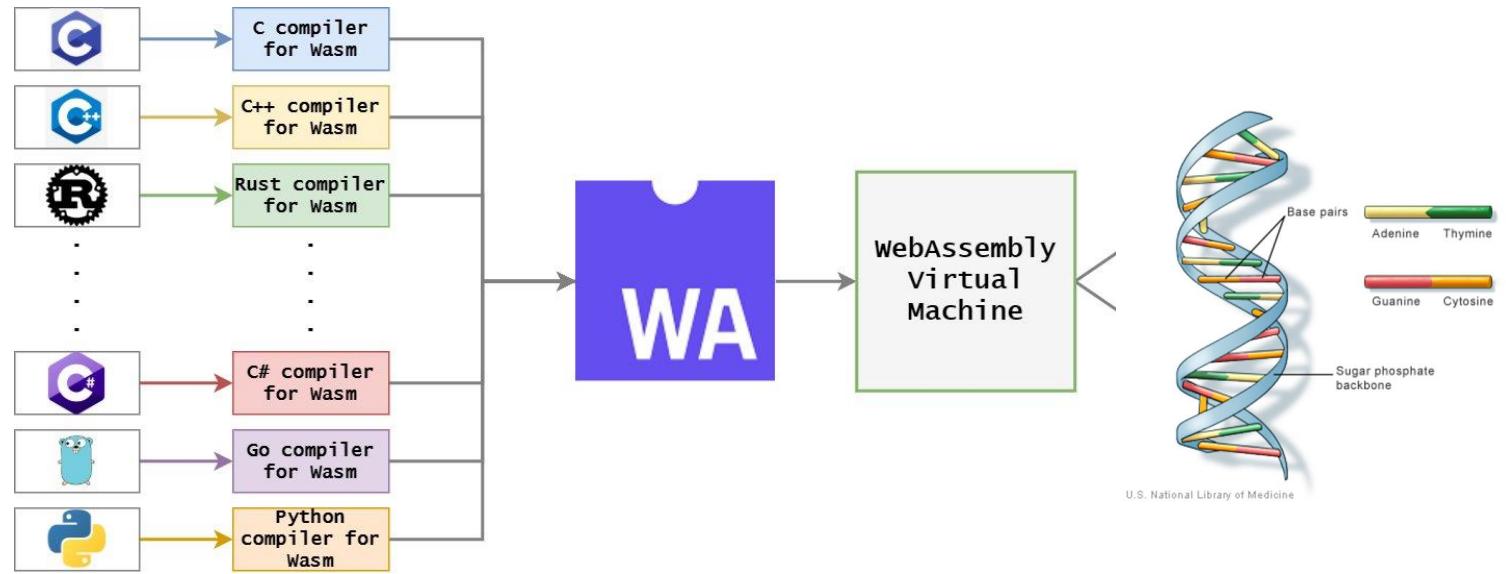
```
npm install phylojs
```



# Lessons learned

- Functionality separate from visualisation
- Let your app have a simple state (ideally JSON serialisable)
- Do the hard science in your language of choice compiled to WASM
  - E.g. I would have used the Phylo Rust/Compact Tree in C++ crate instead of building PhyloJS.

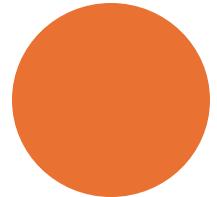
# Bioinformatics Use Cases for WebAssembly



# Wytamma Wirth

<https://biowasm.com>

- Run C/C++ genomics tools in the browser
- Use bioinformatics command-line tools in your web apps with just a few lines of code.  
Powered by WebAssembly.

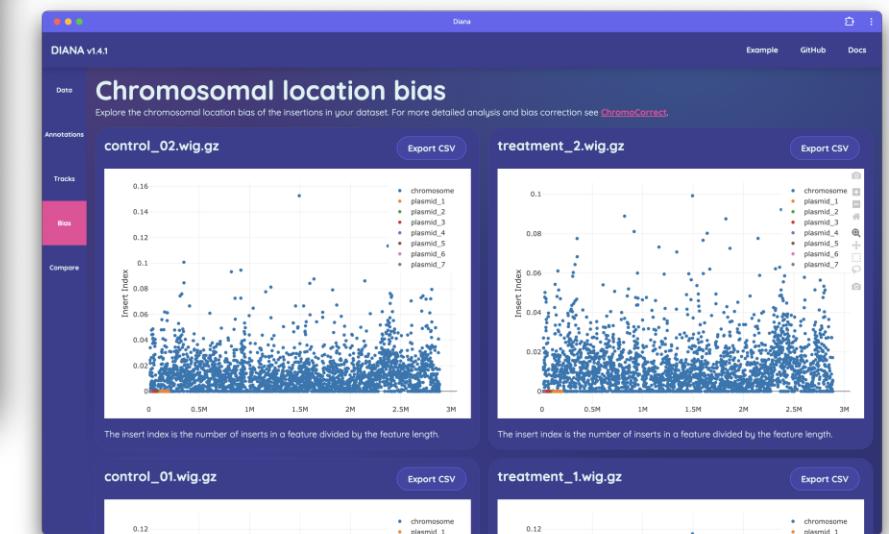
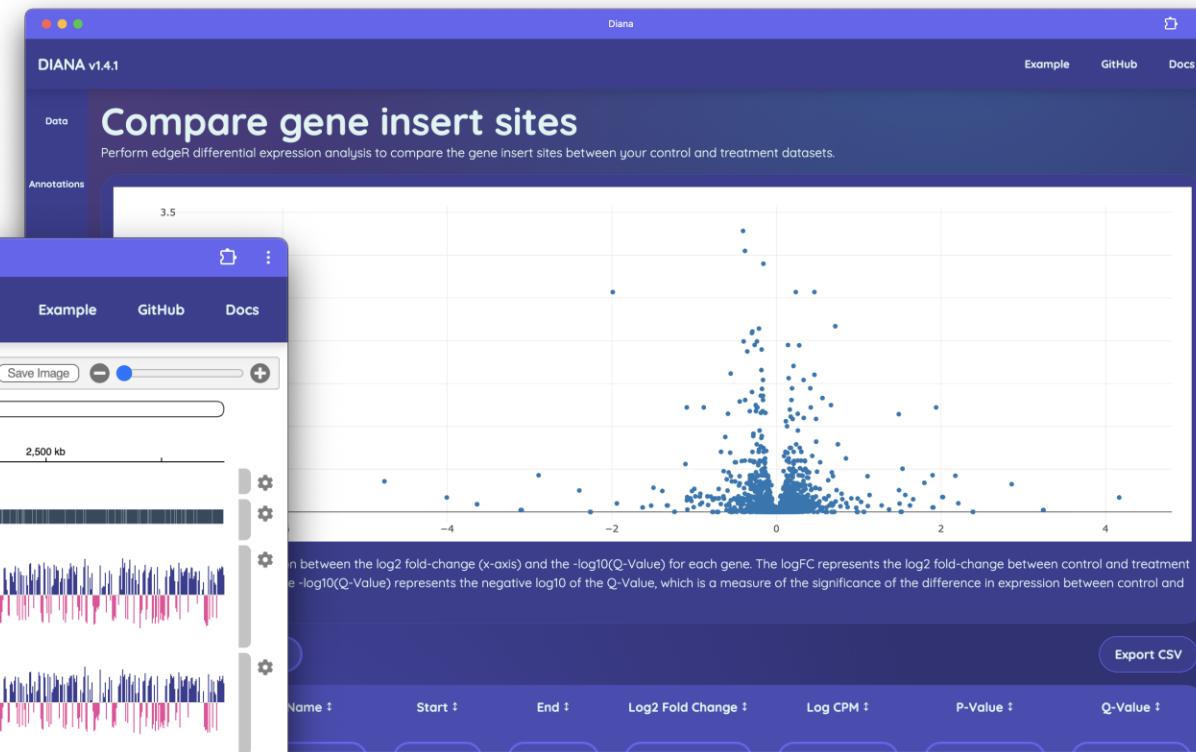


<https://sandbox.bio>

- Interactive bioinformatics tutorials.  
No setup required.

Robert Aboukhalil

<https://diana.cpg.org.au/>



<https://portal.cpg.unimelb.edu.au/>

CPG Portal - JupyterLite

File Edit View Run Kernel Tabs Settings Help

CPG Portal Files

Browse files on the CPG Portal. Use the download button to save files to your Secure Analysis Environment.

File Name	Type	Created	Size
reference	fasta	8d ago	3 MB
M.leprae genomes ...	fasta	8d ago	25 MB
visualization.svg	svg	29d ago	300 KB
annual-precipitatio...	vega	29d ago	3 KB
area_density.vl.json	vega-lite	29d ago	498 B
mist.txt	text	29d ago	882 KB
mist.txt	json	29d ago	882 KB
GCF_009035845.1....	fasta	29d ago	4 MB
mist.json	json	29d ago	289 KB
Ecoli_DERA_Implici...	genbank	29d ago	3 KB
chronumental_met...	csv	1M ago	127 KB
mygenome.tsv	tsv	1M ago	304 KB
barbet-predictions....	csv	1M ago	415 B
reference.fa	fasta	1M ago	5 MB
Hospital outbreak r...	fasta	1M ago	5 MB
Hospital outbreak ...	fasta	1M ago	91 MB
Hospital outbreak ...	tsv	1M ago	1 KB
emboss_needle-i2...	unknown	1M ago	114 B
emboss_needle-i2...	unknown	1M ago	114 B
mutant_R2.fastq_2...	fastq	1M ago	4 MB
mutant_R1.fastq_1....	fastq	1M ago	4 MB
filter.aln	unknown	1M ago	2 KB
example.fa	fasta	1M ago	35 B
assemblies (group)	fasta	1M ago	11 MB

Launcher GCF\_009035845.1.fna Untitled.ipynb

Notebook Python (Pyodide)

```
[3]: with open("GCF_009035845.1.fna") as f:
    for line in f:
        if not line.startswith(">"):
            continue
        print(line)

>NZ_CP045110.1 Acinetobacter baumannii strain ATCC 19606 chromosome, complete genome
>NZ_CP045108.1 Acinetobacter baumannii strain ATCC 19606 plasmid p1ATCC19606, complete sequence
>NZ_CP045109.1 Acinetobacter baumannii strain ATCC 19606 plasmid p2ATCC19606, complete sequence
```

Simple Python (Pyodide) | Idle Mode: Command Ln 1, Col 1 Untitled.ipynb 0



# <https://clades.nextstrain.org/>

Nextclade

Start ▶ Dataset ▶ Results ▶ Tree ▶ Export

Done. Total sequences: 165. Succeeded: 165

#	i	Sequence name	QC	Clade	Pango lineage (Nextclade)	WHO name	Mut.	non-ACGTN	Ns	Cov.	Gaps	Ins.	FS	SC
0	0	OY754687	N M P C F S	23B	XBB.1.16.11		104	2	160	99.5%	56	0	0	0 (1)
1	1	USA/CA-LACPHL-AY03266/2023	N M P C F S	23F	EG.5.1.13		109	0	1229	95.1%	39	0	0	0 (1)
2	2	OY754528	N M P C F S	23E	GE.1		69	14	11269	62.3%	0	0	0	0
3	4	USA/CA-LACPHL-AY03267/2023	N M P C F S	23F	EG.5.1.16		106	0	0	99.3%	30	0	0	0 (1)
4	3	OY754651	N M P C F S	23F	EG.5.1		103	2	3400	88.6%	9	0	0	0 (1)
5	6	USA/WA-UW-23102346451/2023	N M P C F S	recombinant	XCR	recombinant	110	0	0	98.8%	30	0	0	0 (1)
6	5	OY754526	N M P C F S	23E	GJ.2		98	0	1944	93.5%	9	0	0	0
7	7	USA/WA-UW-23102330989/2023	N M P C F S	23F	HV.1		108	0	0	98.8%	30	0	0	0 (1)
8	8	USA/CA-LACPHL-AY03271/2023	N M P C F S	23G	XBB.1.5.70		98	0	756	96.7%	30	0	0	0 (1)
9	9	OY754632	N M P C F S	23I	JN.3.3	Omicron	114	2	185	99.4%	3	0	0	0
10	10	USA/CA-LACPHL-AY03247/2023	N M P C F S	23B	JF.4		110	0	129	98.8%	32	0	0 (1)	0 (1)
11	11	USA/CA-LACPHL-AY03246/2023	N M P C F S	23A	HR.1.1		109	0	1	99.2%	30	0	0	0 (1)
12	12	OY754626	N M P C F S	23F	EG.5.1.3		106	2	143	99.5%	9	0	0	0 (1)
13	13	USA/CA-LACPHL-AY03246/2023	N M P C F S	23C	DV.7.1.4	Omicron	108	0	1398	94.6%	27	0	0	0
14	14	OY754681	N M P C F S	23B	JF.3		110	2	83	99.7%	56	0	0	0 (1)
15	15	OY754673	N M P C F S	recombinant	XCH.1	recombinant	113	5	142	99.5%	9	0	0	0 (1)
16	16	USA/LA-EVTL20384/2023	N M P C F S	23A	JD.1.1		105	0	0	99.9%	21	0	0	0 (1)
17	17	USA/NY-PBRI-NYC20026/2023	N M P C F S	23F	EG.5.1.4		108	0	0	99.6%	56	0	0	0 (1)
18	18	OY754523	N M P C F S	23F	EG.5.1.3		107	2	143	99.5%	9	0	0	0 (1)
19	19	USA/CA-LACPHL-AY03248/2023	N M P C F S	23D	FL.2.5		108	0	1	99.2%	30	0	0	0 (1)

Genome annotation ?

Nextclade (c) 2020-2025 Nextstrain developers

BZ

EN

version 3.18.0 (commit: cae18ab, branch: release)

Nextclade

Start ▶ Dataset ▶ Results ▶ Tree ▶ Export

Done. Total sequences: 165. Succeeded: 165

Color By Clade

Filter Data Type filter query here... Currently selected filter categories: 1 x Node type

Tree Layout RECTANGULAR RADIAL UNROOTED SCATTER Focus on Selected Branch Labels clade Show all labels Tip Labels Sample Name

Auspice

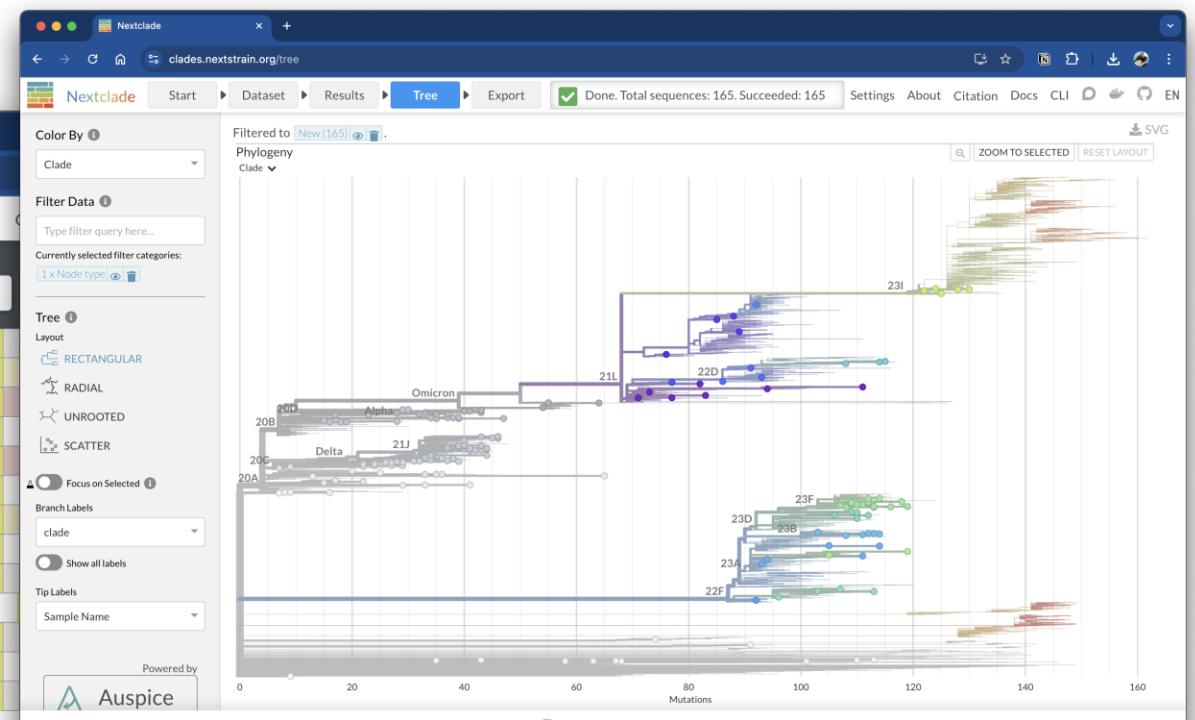
Filtered to New(165). ZOOM TO SELECTED RESET LAYOUT

Phylogeny Clade

Nextclade (c) 2020-2025 Nextstrain developers

BZ EN

version 3.18.0 (commit: cae18ab, branch: release)



# Killer Features

WebAssembly is not a silver bullet, but it opens a new niche for software development i.e., real software in the browser.

Reasons why people are excited about WASM:

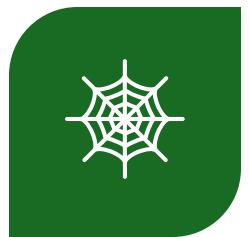
- Portability (works across platforms and browsers).
- Security (sandboxed execution).
- Programming language interoperability
- Deterministic
- FREE HOSTING!!!!



# WebAssembly will not solve everything - There are many limitations



CLIENT-SIDE IS INHERENTLY  
OPEN-SOURCE (NO  
SECRETS)



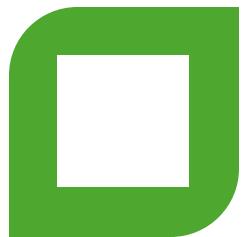
BROWSERS HAVE A DEFAULT  
HIGH LEVEL OF SECURITY  
(NO ARBITRARY FILE  
ACCESS)



LIMITED MEMORY (MAX  
4GB/16GB IN BROWSERS)



NO DATA PERMEANCE (NO  
CENTRAL SERVER OR  
DATABASE)



MULTITHREADING HAS  
LIMITED SUPPORT / IS HARD  
TO SETUP

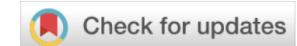
# Toblerone



"If I had my time again"--  
this workshop has been a chance to have my time again

[F1000Research](#)

F1000Research 2023, 12:130 Last updated: 13 NOV 2025



RESEARCH ARTICLE

## Toblerone: detecting exon deletion events in cancer using RNA-seq

[version 1; peer review: 2 approved]

Andrew Lonsdale  <sup>1,3</sup>, Andreas Halman <sup>1,3</sup>, Lauren Brown <sup>2,4,5</sup>, Hansen Kosasih  <sup>2</sup>, Paul Ekert <sup>1-5</sup>, Alicia Oshlack  <sup>1,3,6</sup>

<sup>1</sup>Peter MacCallum Cancer Centre, Parkville, VIC, 3052, Australia

<sup>2</sup>Murdoch Children's Research Institute, Parkville, VIC, 3052, Australia

<sup>3</sup>Sir Peter MacCallum Department of Oncology, University of Melbourne, Parkville, VIC, 3010, Australia

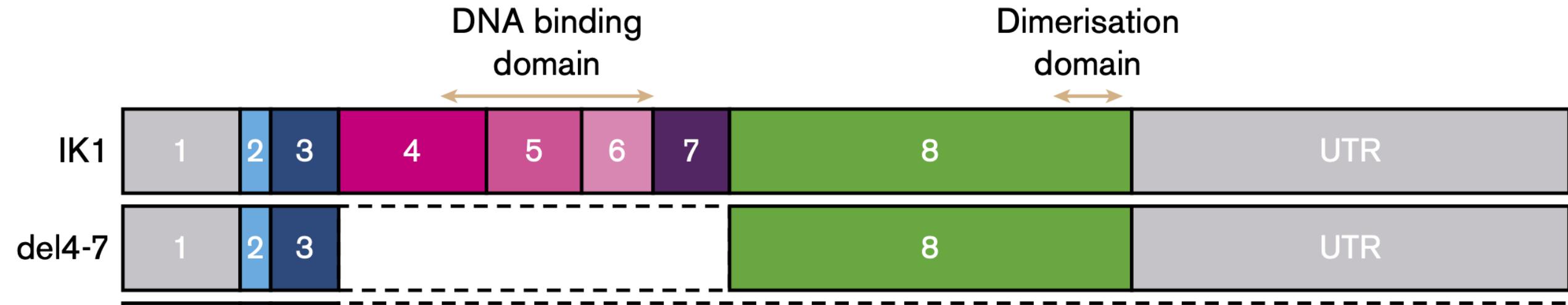
<sup>4</sup>Children's Cancer Institute Australia, Sydney, NSW, 2052, Australia

<sup>5</sup>School of Women's and Children's Health, UNSW Sydney, Sydney, NSW, 2052, Australia

<sup>6</sup>School of Mathematics and Statistics, University of Melbourne, Parkville, VIC, 3010, Australia

# Toblerone

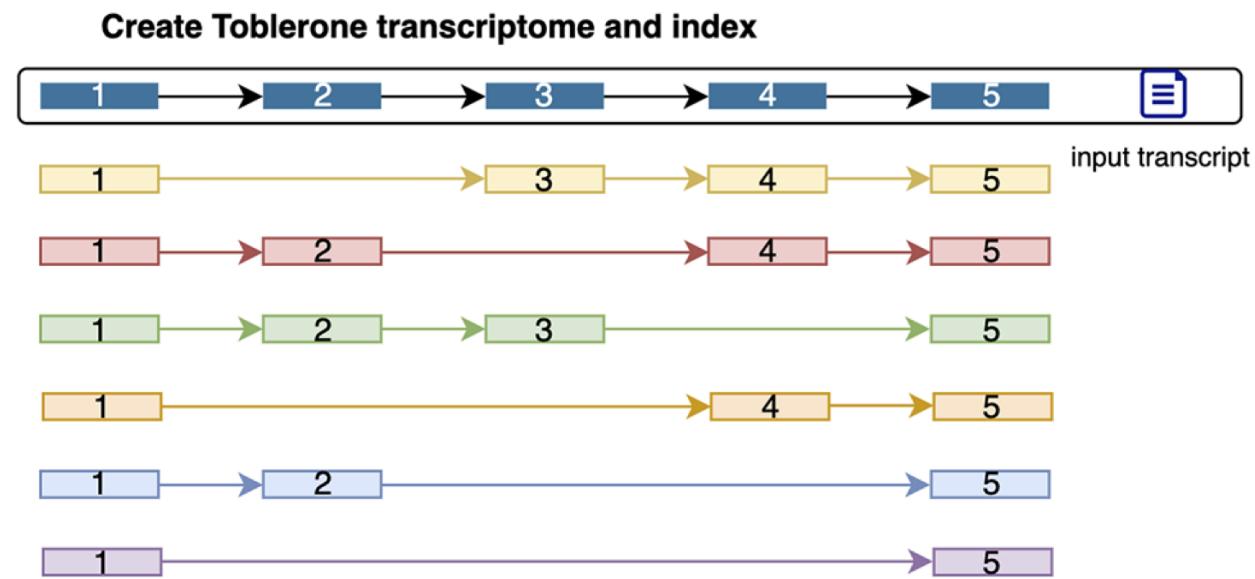
Toblerone is a RNA-seq **pseudoalignment** approach that uses a reference of **only exon deletion transcripts** and counts reads unique to those transcripts to identify exon deletions in cancer.



The del4-7 deletion Ikaros (IKZF1) is clinically relevant in B-ALL

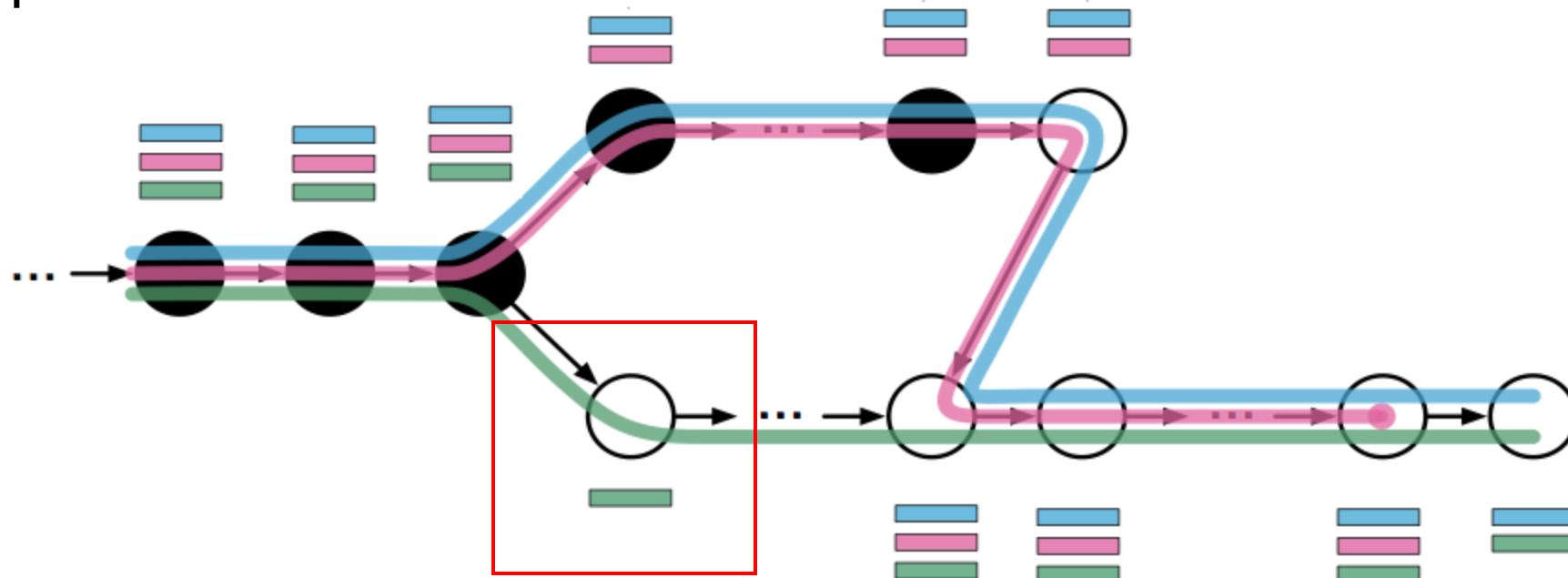
# Toblerone

Toblerone generalises the problem to any exon deletion for a given gene – and generates each potential exon deletion independently



# Simplified pseudoalignment

Because of the way the transcriptome is constructed, exon deletions correspond to these **nodes** on a de Bruijn graph of that transcriptome



# Simplified quantification

Toblerone only counts the reads that map to those unique equivalence classes, scaled with the size of the gene and read length, and relative to the gene expression

$$\text{ScaledProportion} = \frac{\text{Unique Count}}{\text{Total Count}} \cdot \frac{\text{Gene Length}}{\text{Read Length}}$$

# Mapping

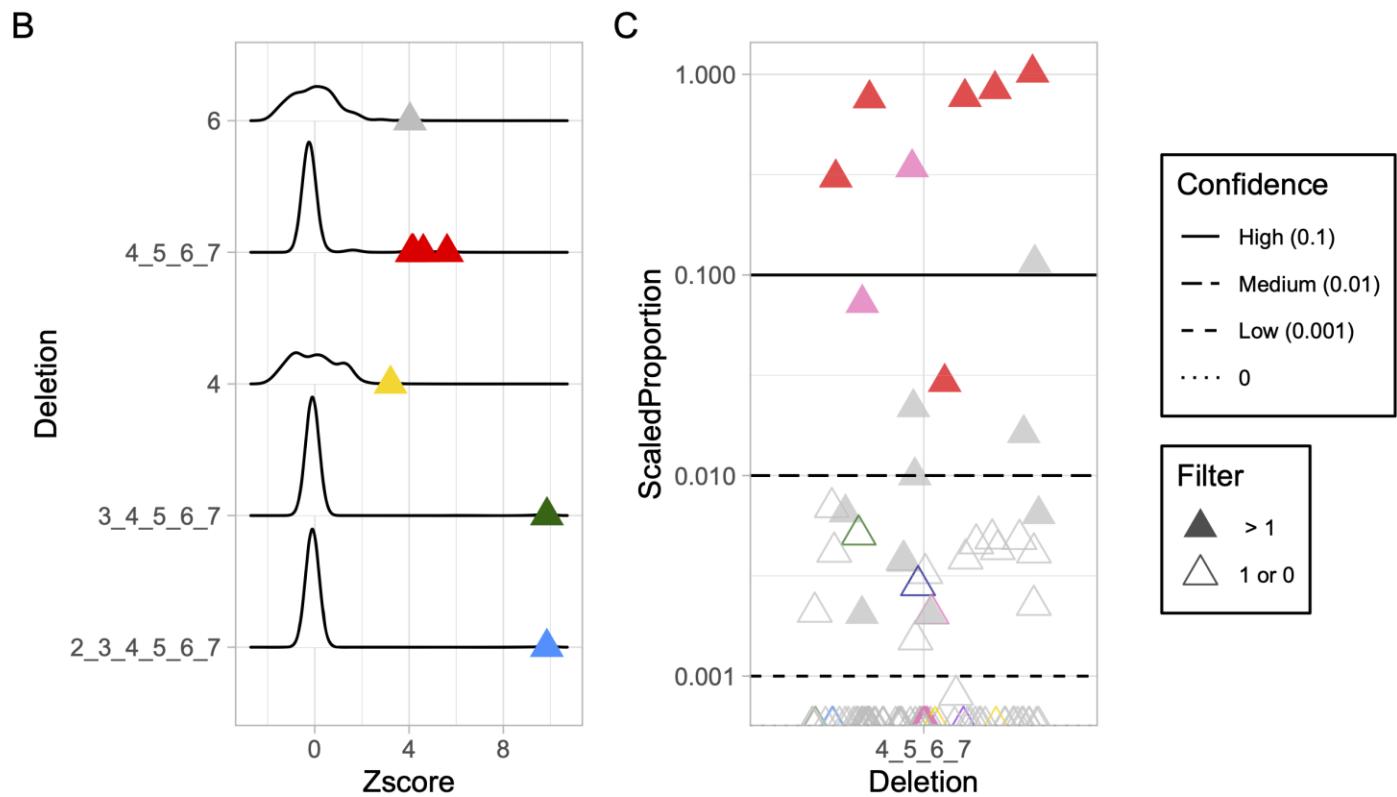
- Given the de Bruijn index for a gene, e.g. IKZF1, reads are mapped and the scaled proportions for each of the possible exon deletions are calculated
- Called the command line interface `tinyt`

```
$ ./tinyt map -i ikzf1_deletions.idx input.fastq
```



# Simplified quantification

- Cohorts are used to find outliers and create clinical thresholds.
- It performs well on IKZF1 using true-positive deletions



# CLI vs App

## Toblerone

- Custom pseudoaligner (`tinyt`)
- Command line tool – `index` and `map`
- FASTQ input
- Generic approach
- Apply to many samples in parallel
- Output CSV



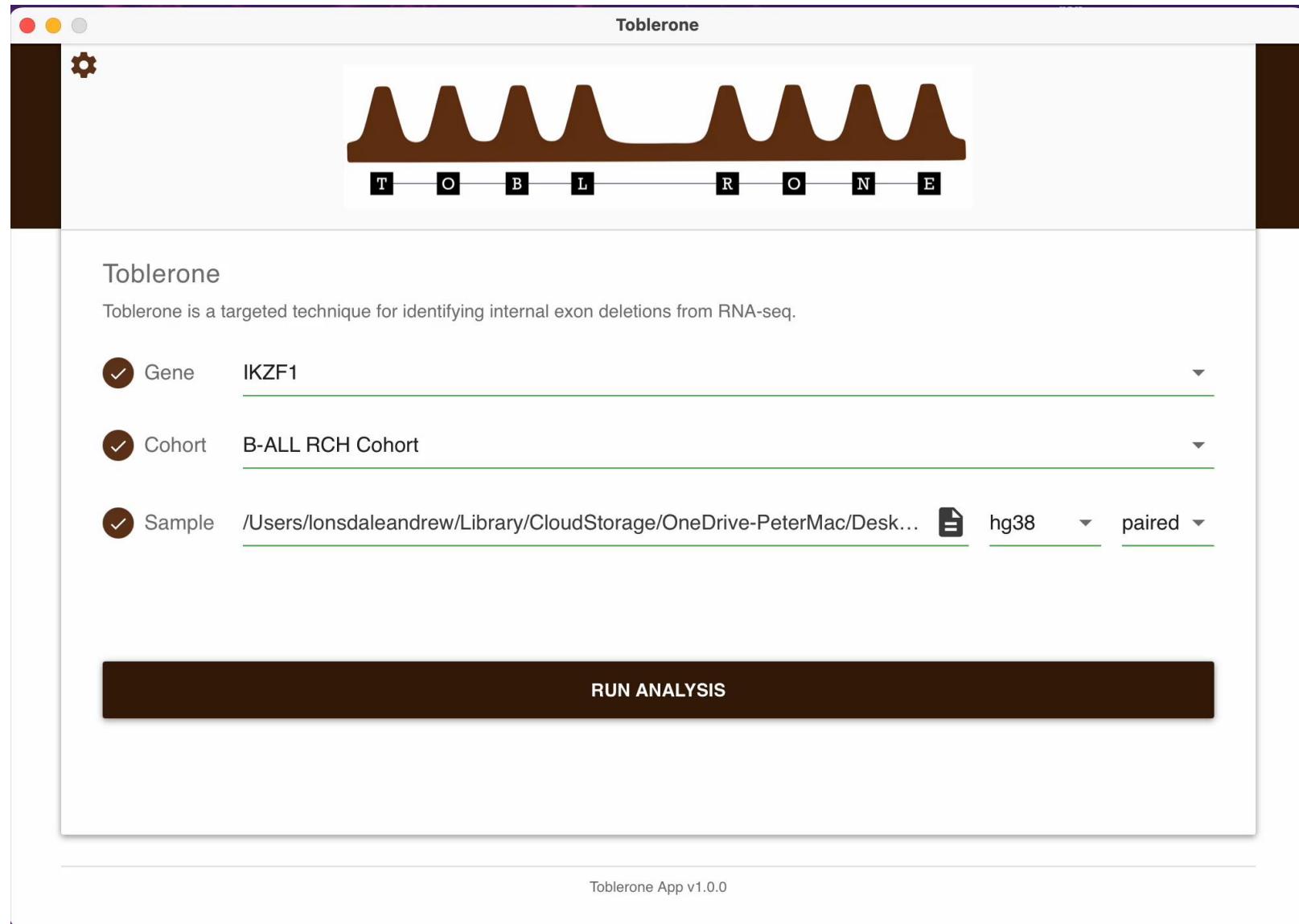
## Toblerone App

- Desktop app for clinical user
- **Calls `tinyt` binary locally**
- IKZF1 index provided
- Pre-aligned genome BAM input
- Includes RCH cohort as reference
- Mac, Windows, Linux



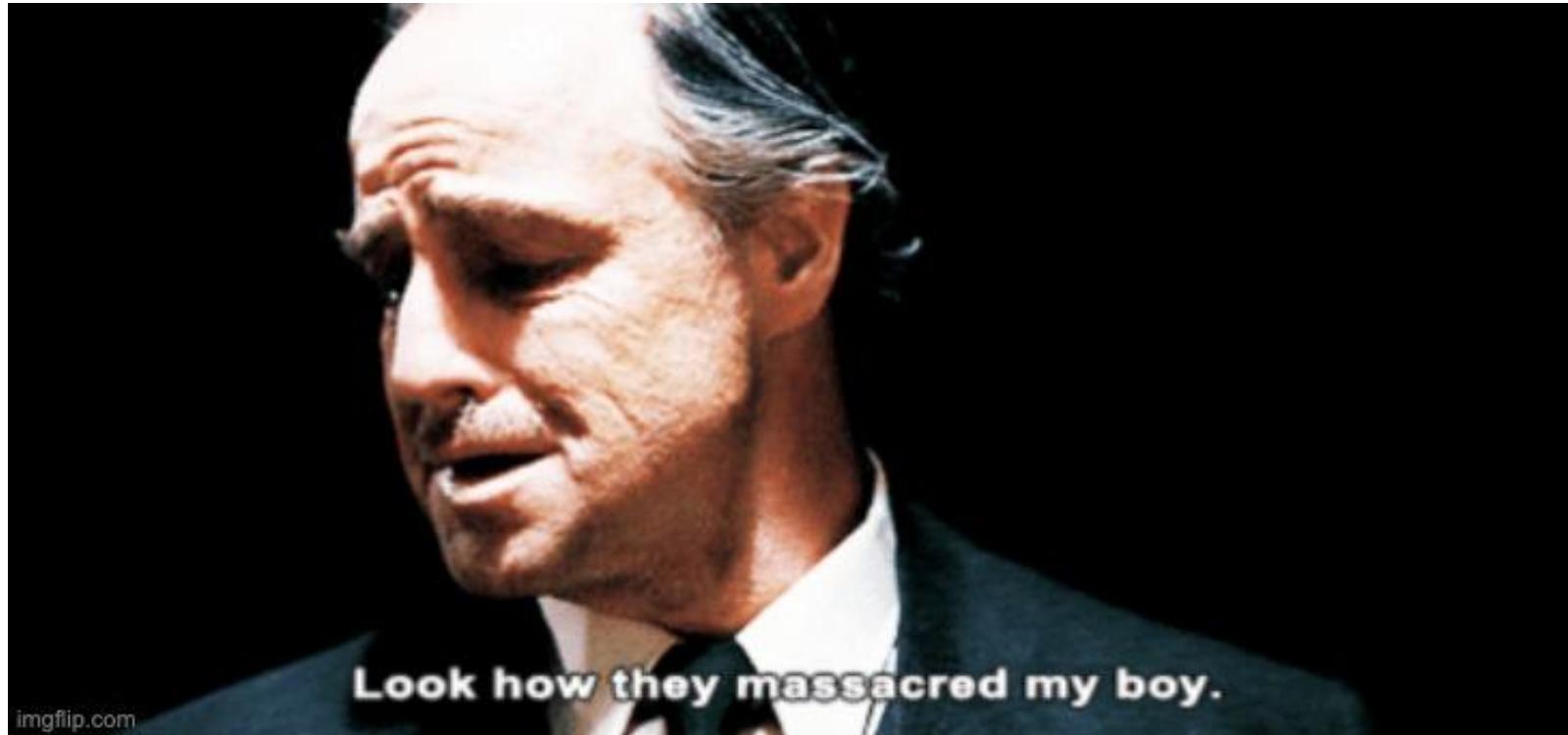
# Compromises

- Lost ggplot
- New JS plotting with HighCharts library
- Operating system install issues, especially on WSL2
- Custom code to extract reads from BAM files



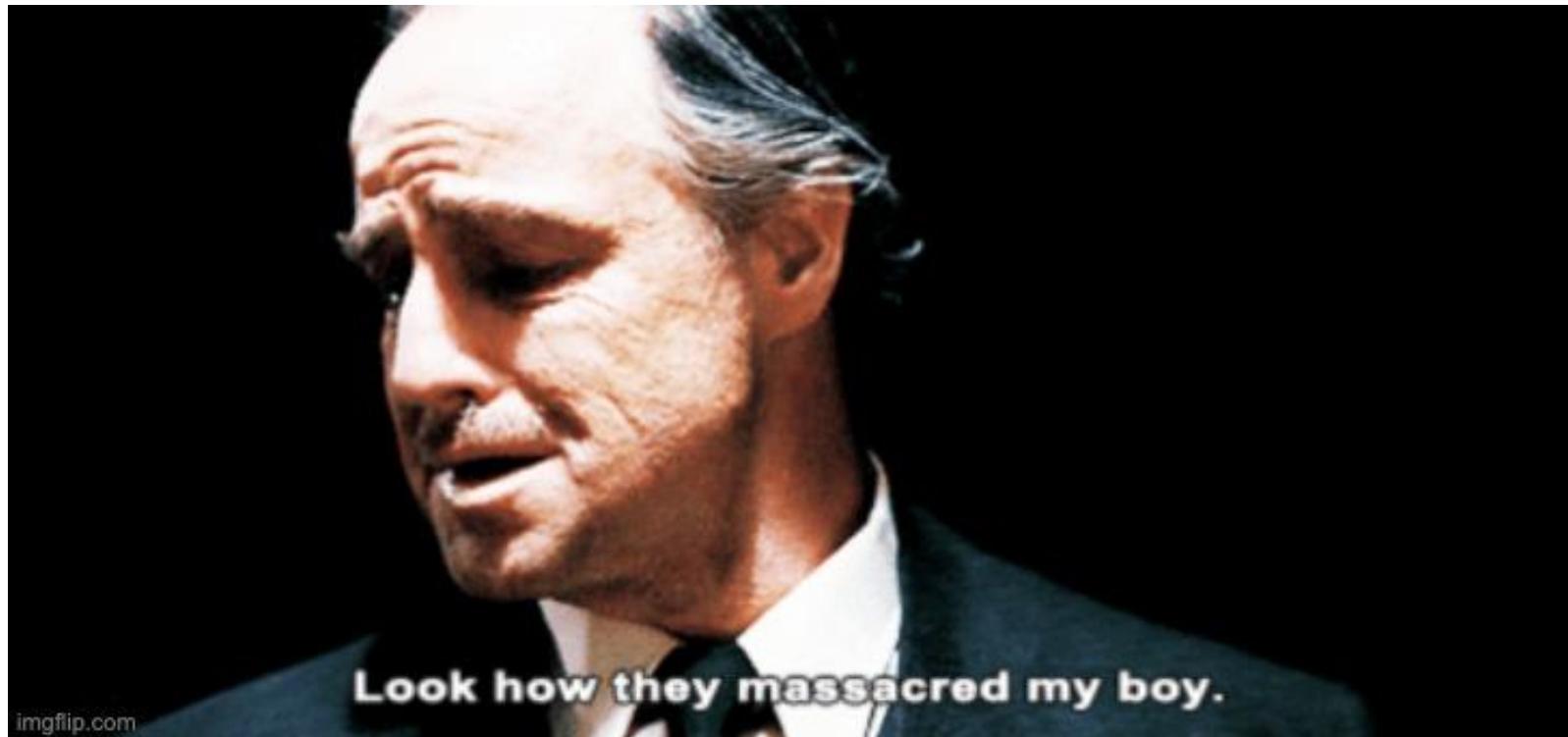
# Compromises

- Lost ggplot
- New JS plotting with HighCharts library
- Operating system install issues, especially on WSL2
- Custom code to extract reads from BAM files



# Compromises

- Successful...but it never entirely satisfied
- Published though, so time to move on...



# Web Assembly

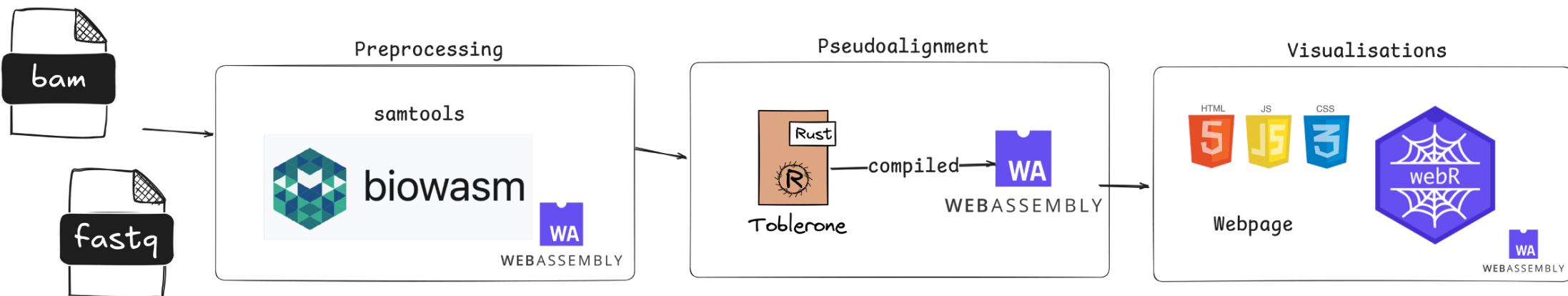
WA



# Toblerone WASM



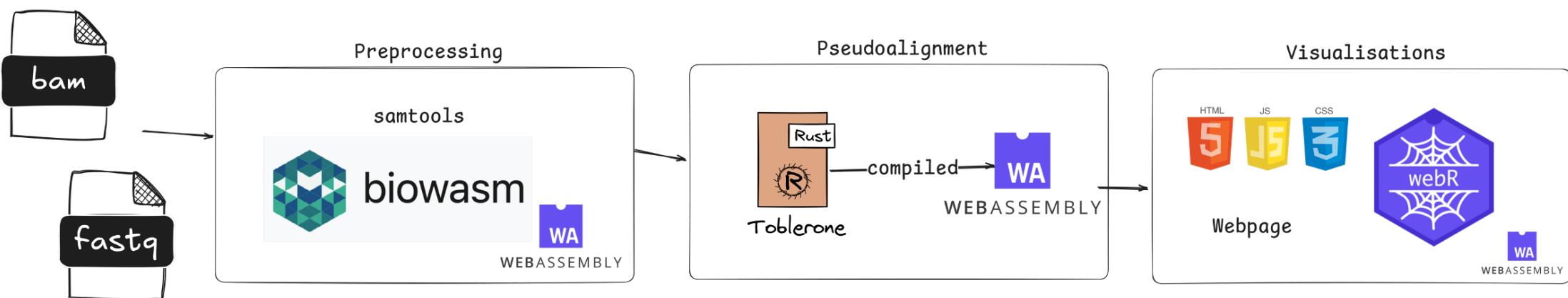
- Key benefits:
    - Core Toblerone code written in Rust can be compiled directly to WASM
    - Read extraction can use a WASM version of samtools over custom Python scripts (more on Biowasm later in workshop)
    - Original plot style can be restored using adapted code from paper in the WASM version of R, WebR



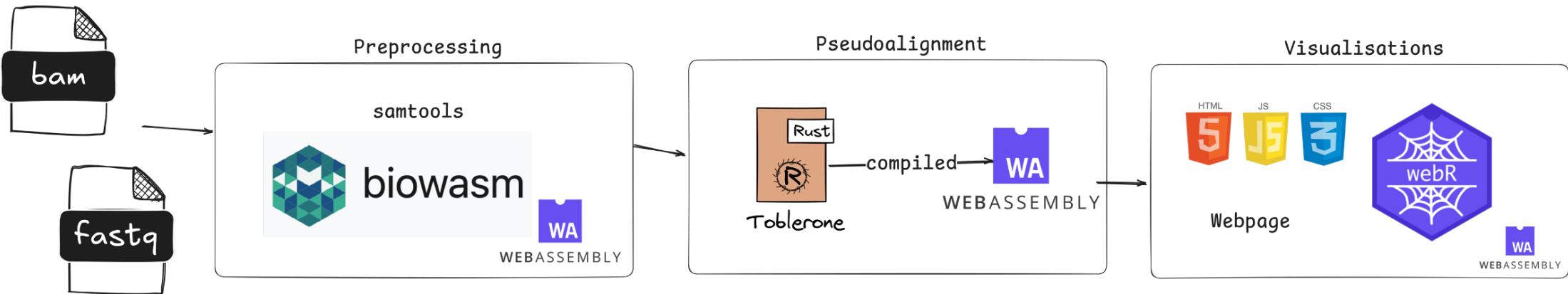
# Toblerone WASM



- Extra benefits:
  - Users avoid installing software.
  - Avoids most cross-platform compatibility issues.
  - Distributing new and updated reference indexes and cohorts become simpler.



# Toblerone WASM

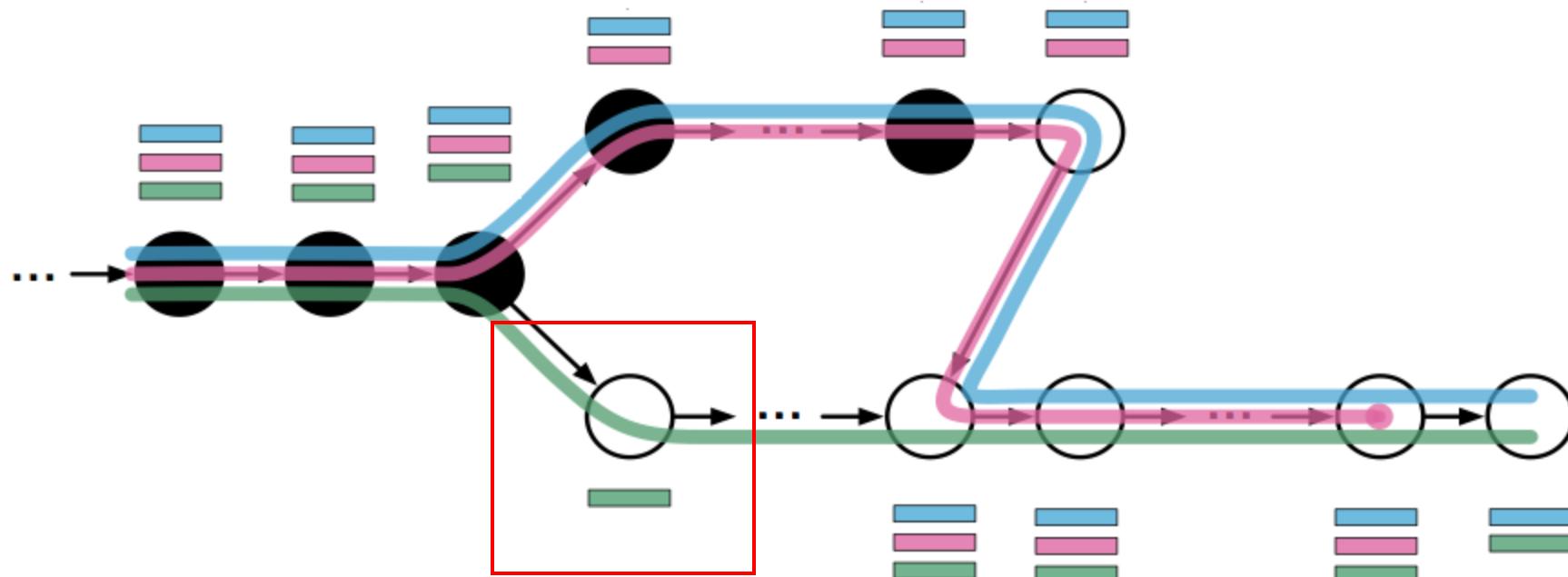


# Conversion lessons

- You can't do everything (or at least, not in the same way!)
  - e.g. threading when choosing Biowasm
  - Error: Result::unwrap() on an Err value: Os { code: 6, kind: WouldBlock,message: "Resource temporarily unavailable" }
  - Each call to Biowasm uses a web worker as a thread
  - Using threads in the same way as a local machine target platform won't work directly
  - Solution: avoid threads in indexing and mapping
  - Speaking of indexing...

# Conversion lessons

- You can't do everything (or at least, not in the same way!)
  - e.g. tinyt index files are essentially de Bruijn graphs saved to file



# Conversion lessons

- You can't do everything (or at least, not in the same way!)
  - e.g. tinyt index files are essentially de Bruijn graphs saved to file
  - Rust library for the graph relies on 64-bit pointers
  - Index creation in WASM going to be difficult
  - Do we even need to?

# Platform agnostic index

- Once the graph is made, do we need it?
- For a read, we know that we only want to count unique Equivalence Classes (ECs) that match one possible deletion transcript
- Making indexes offline, and making them lookup tables of k-mer -> equivalence class makes it simpler and more portable .. I think

# Add --wasm option and a dual Index approach

```
37 + tinyt index [--num-threads=<n>] [--wasm] --index=<index> <ref-fasta>
38 + tinyt map [--num-threads=<n>] [--read-length=<r>] [--trim-size=<t>] [--skip-trim] [--mismatch=<m>] [--output=<file>] [--wasm]
    --index=<index> <reads-fastq> [<reads-pair-fastq>]
```

43	+ -w --wasm	Create or read index in WASM compatible format
----	-------------	--

```
16 - build_index::build_index,
```

```
18 + build_index::{build_index,export_wasm_index,WasmRuntimeIndex, WasmIndex,IndexLike},
```

# IndexLike

```
202 +         let index_box: Box<dyn IndexLike> = if args.flag_wasm {
203 +             // read the compact WasmIndex (must implement Deserialize)
204 +             let wasm_idx: WasmIndex = utils::read_obj(&args.flag_index)?;
205 +             Box::new(WasmRuntimeIndex::from_wasm_index(wasm_idx))
206 +         } else {
207 +             // read the full native Pseudoaligner (concrete type)
208 +             let native_idx: Pseudoaligner<config::KmerType> = utils::read_obj(&args.flag_index)?;
209 +             // let native_idx = utils::read_obj(&args.flag_index)?;
210 +
211 +             Box::new(native_idx)
212 +         };
213
```

# IndexLike

```
514 - pub fn match_read<K: Kmer + Sync + Send>(optional: Option<(Vec<u32>, usize, usize, usize)>, seq  
      trim: bool, trimsize: usize, mismatchsize: usize, seqlength: usize, index: &Pseudoaligner<K> )  
      Option<(bool, bool, String, Vec<u32>, usize, usize, bool, usize)> {  
541 + pub fn match_read(optional: Option<(Vec<u32>, usize, usize, usize)>, seq: &String, record_id:  
      usize, mismatchsize: usize, seqlength: usize, index: &dyn IndexLike ) -> Option<(bool, bool, Str  
      usize, usize, bool, usize)> {  
515 542 27 + use std::collections::{HashMap, HashSet};
```

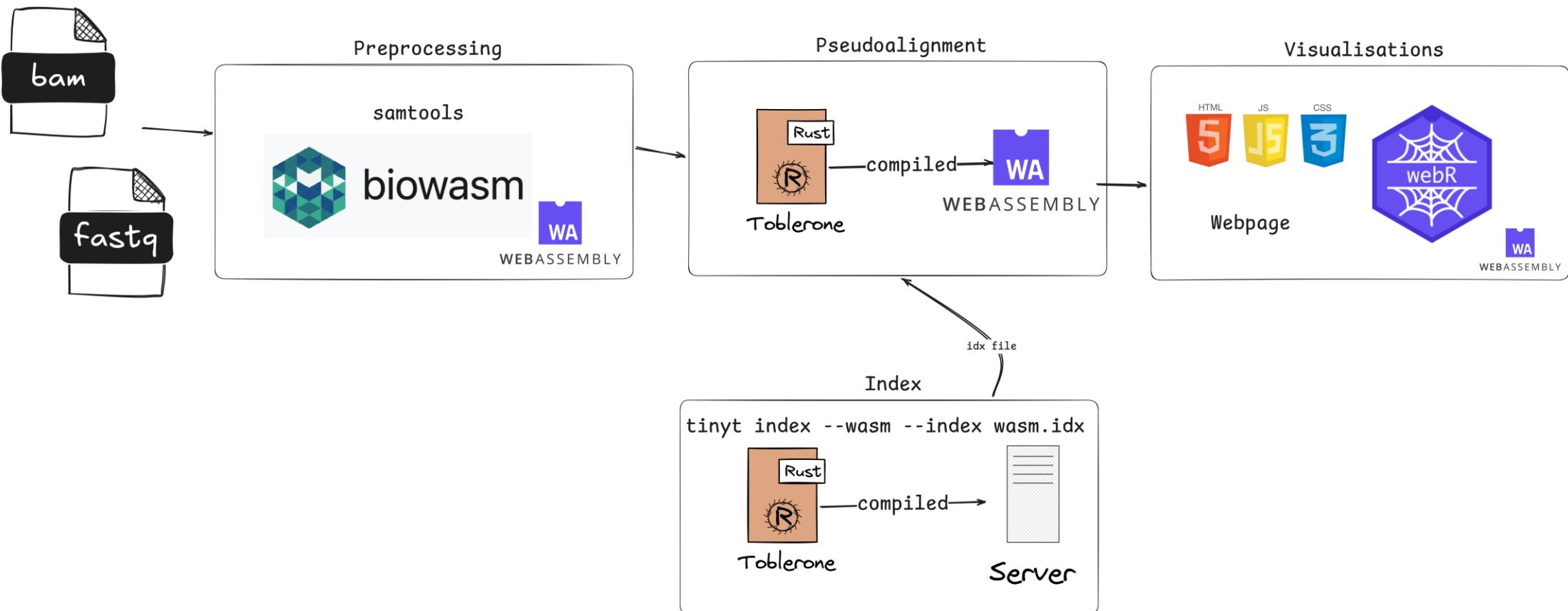
```
133 + pub trait IndexLike: Sync {  
134 +     fn map_read(&self, read_seq: &DnaString, mismatch_size: usize) -> Option<(Vec<u32>, usize, usize, usize)>;  
135 +     fn tx_names(&self) -> &Vec<String>;  
136 +     fn tx_gene_mapping(&self) -> &HashMap<String, String>;  
137 +     fn gene_length_mapping(&self) -> &HashMap<String, usize>;  
138 + }
```

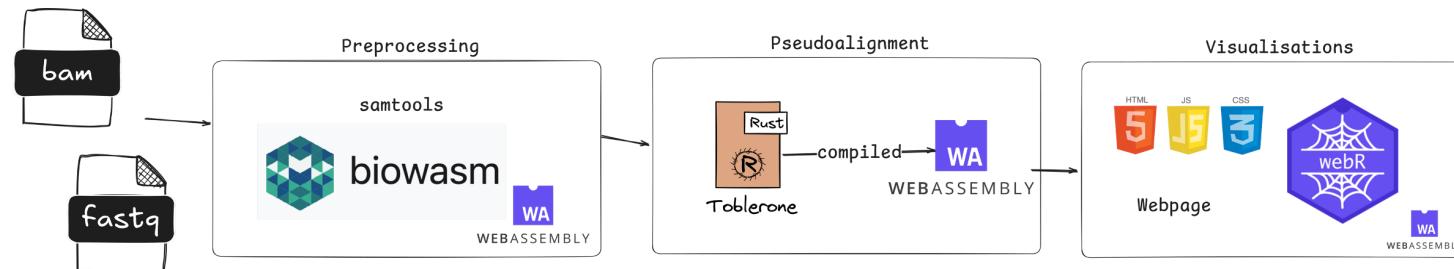
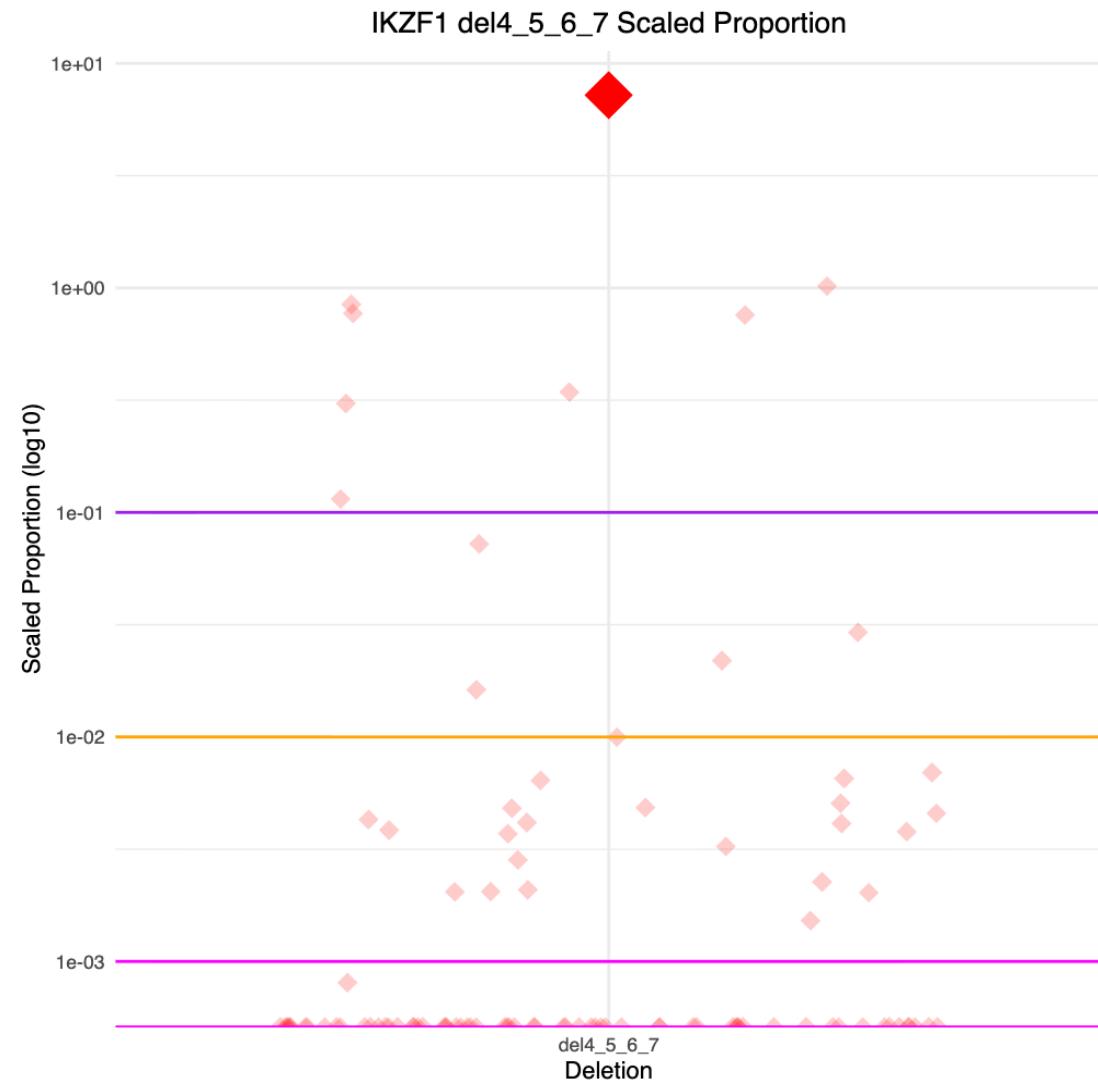
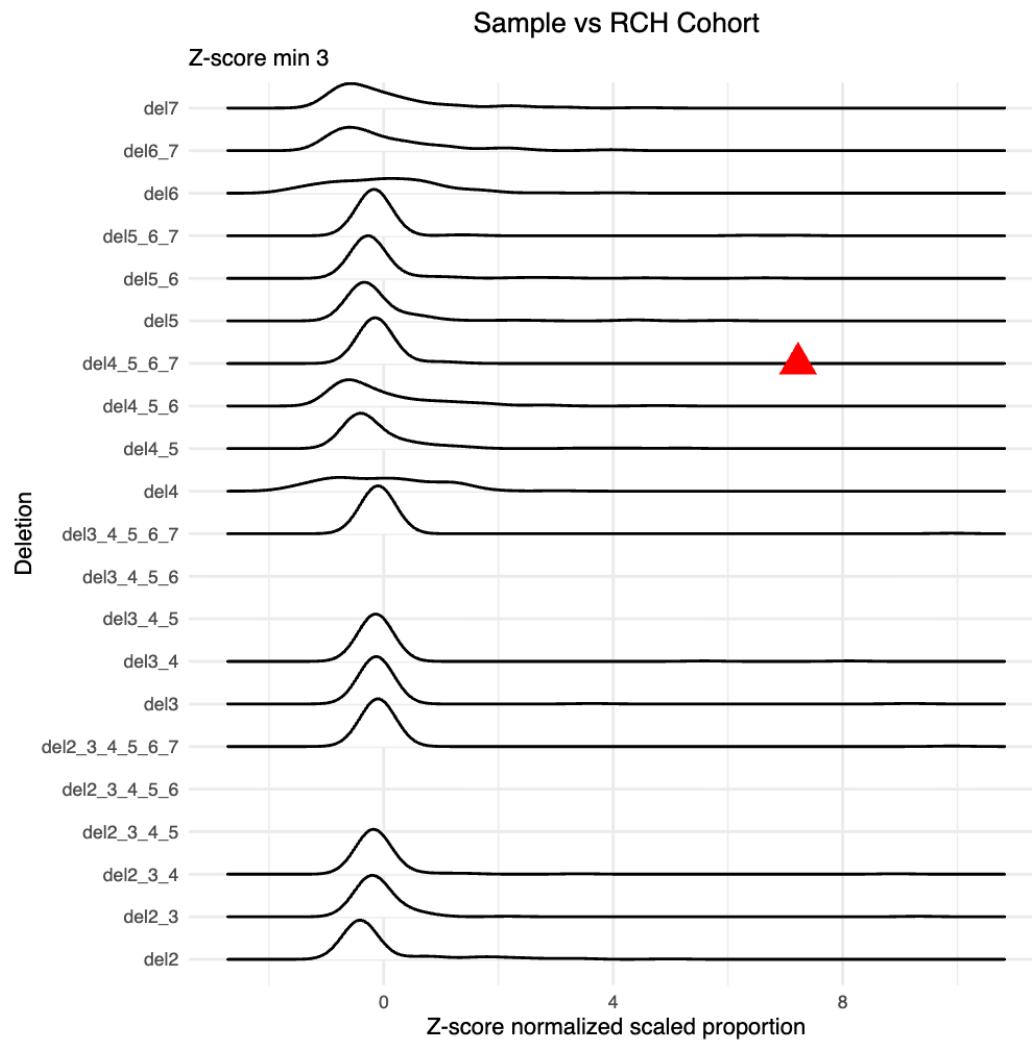
# IndexLike

```
345 + impl IndexLike for WasmRuntimeIndex {  
346 +     fn map_read(&self, read_seq: &DnaString, _allowed_mismatches: usize) -> Option<(Vec<u32>, usize, usize, usize)> {  
347 +         // convert read into canonical k-mer u64s (utils::kmers_to_u64_vec must match exporter encoding)  
348 +         let kmers = crate::utils::kmers_to_u64_vec(read_seq, self.k as usize);  
349 +         if kmers.is_empty() {  
350 +             return None;  
351 +         }
```

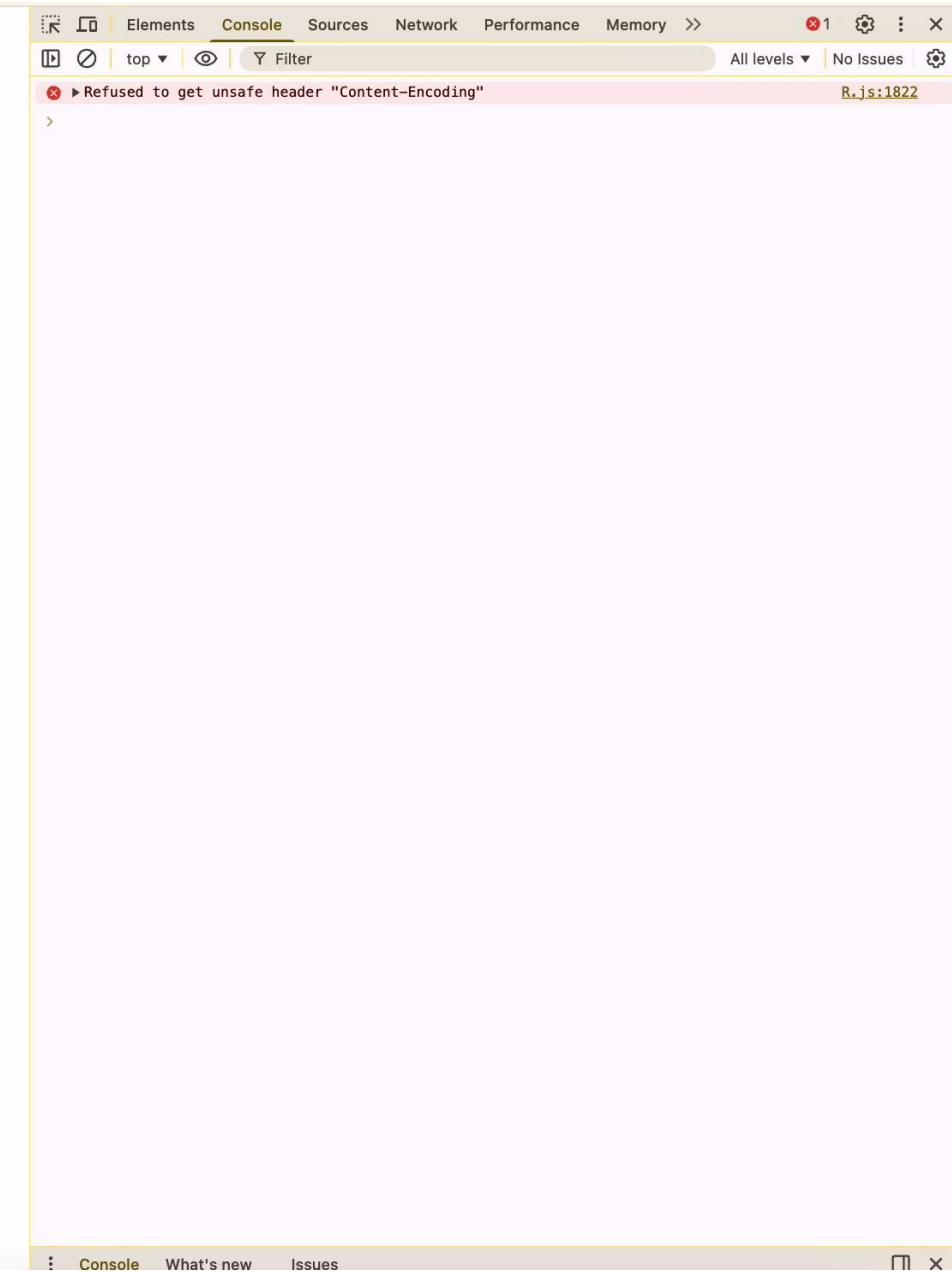
```
415 + // Exporter: walk the de Bruijn graph and collect all kmers -> node+offset  
416 + pub fn export_wasm_index<K: Kmer>(al: &Pseudoaligner<K>) -> WasmIndex {  
417 +     let mut kmers = Vec::new();  
418 +     let klen = K::k() as u8;
```

# Toblerone WASM





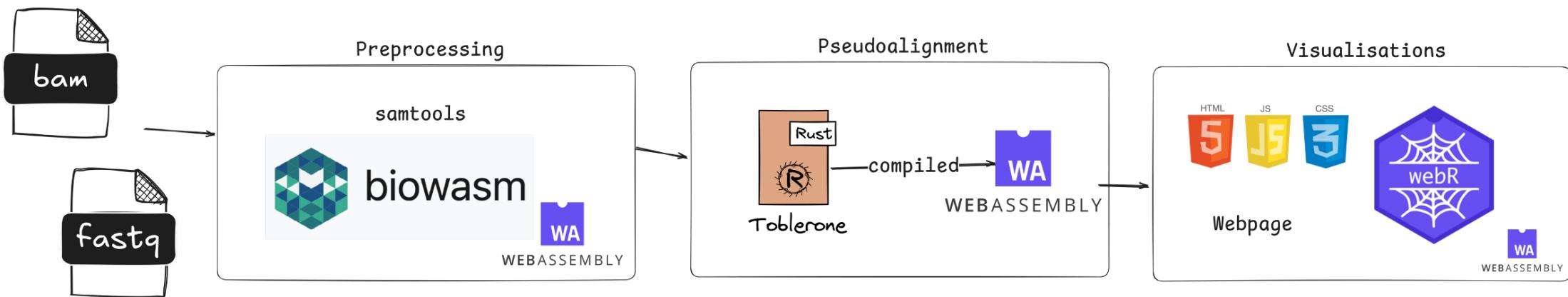
## IKZF1 – WebR ggplot PNG demo



# Summary



- Using WebAssembly (WASM), we created a client-side web browser version of Toblerone that reduces maintenance effort, and enhances accessibility and privacy.
- It is under development and available at <https://github.com/Oshlack/TobleroneWASM>

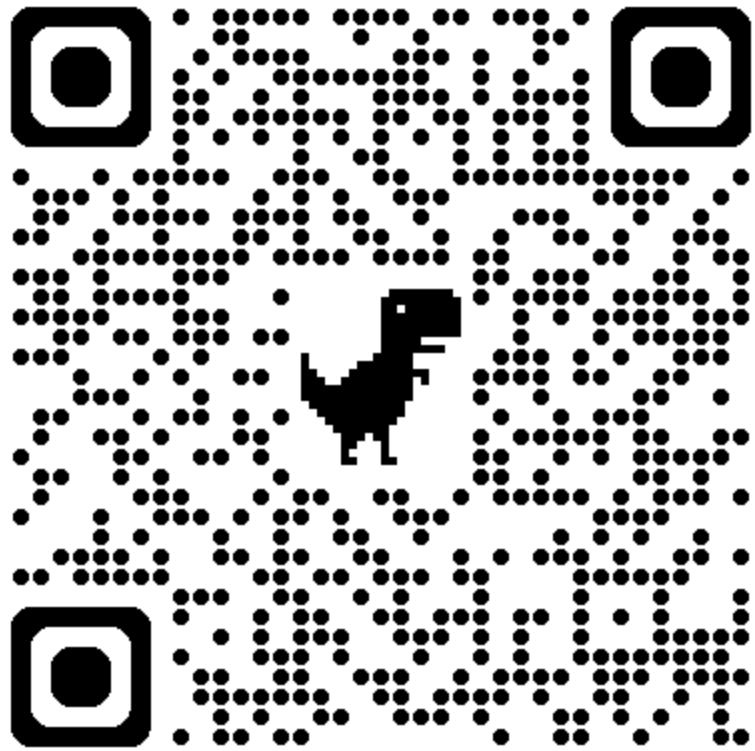


# Conclusion



- Single code base for multiple platforms ideal
  - Currently gated with --wasm flag but could do at compile time
  - I think there is a bug in the current TobleroneApp JS – found when able to use original R code again to visualise
- WASM is awesome but conversion is hard!
- If I had my time again (again) - develop with one eye on WASM
  - make choices about what is a universal feature, what might not be available in WASM version of tool

<https://wasmmodic.github.io/>



ABACBS 2025

## Client-side Bioinformatics

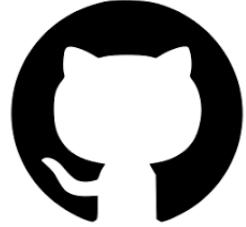
*Building bioinformatics tools for the browser with WebAssembly*

```
~ > csvtk head iris.csv | csvtk pretty
sepal.length  sepal.width  petal.length  petal.width  variety
-----  -----  -----  -----  -----
5.1      3.5       1.4       .2        Setosa
4.9      3          1.4       .2        Setosa
4.7      3.2       1.3       .2        Setosa
4.6      3.1       1.5       .2        Setosa
5         3.6       1.4       .2        Setosa
5.4      3.9       1.7       .4        Setosa
4.6      3.4       1.4       .3        Setosa
5         3.4       1.5       .2        Setosa
4.4      2.9       1.4       .2        Setosa
4.9      3.1       1.5       .1        Setosa
~ >
```

Modified from [sandbox.bio](#)

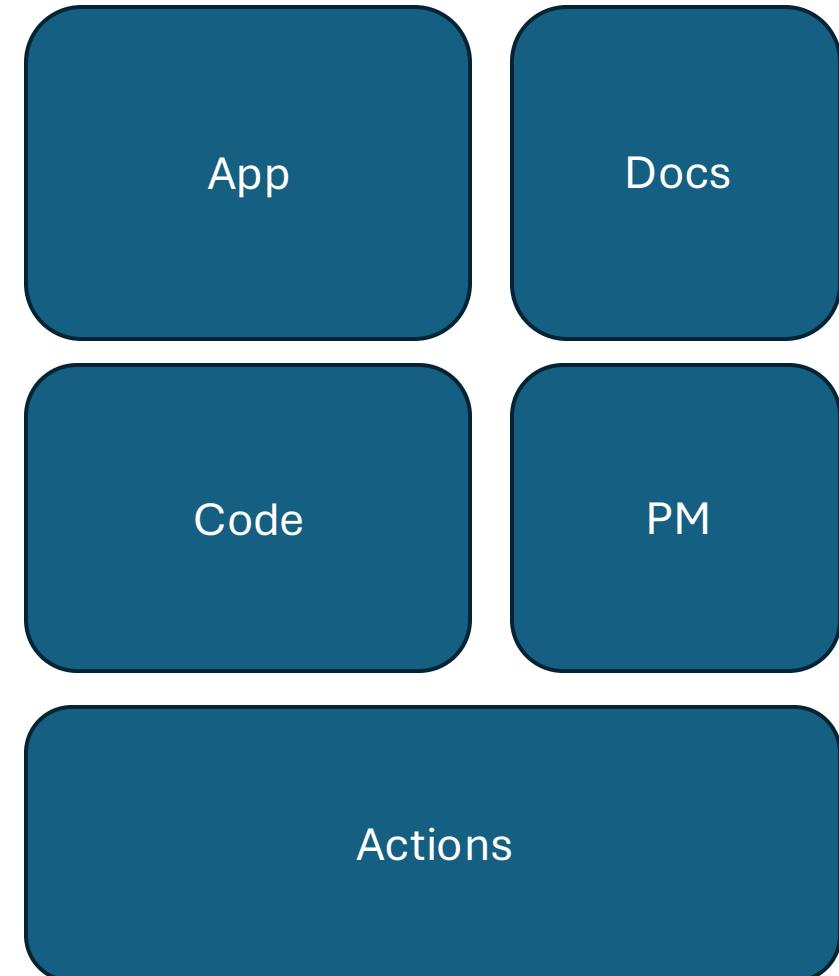
ABACBS Wasm Workshop

# GitHub as Opensource Infrastructure



Github serves as our deployment platform.

We use GitHub Actions to automatically build and deploy the webapp to GitHub Pages whenever we push a new release.



[Preview] README.md — dev

github.dev/github/dev

dev [GitHub]

EXPLORER

DEV [GITHUB]

README.md

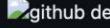
## What is this?

The github.dev web-based editor is a lightweight editing experience that runs entirely in your browser. You can navigate files and source code repositories from GitHub, and make and commit code changes.

There are two ways to go directly to a VS Code environment in your browser and start coding:

- Press the . key on any repository or pull request.
- Swap `.com` with `.dev` in the URL. For example, this repo <https://github.com/github/dev> becomes <http://github.dev/github/dev>

Preview the gif below to get a quick demo of github.dev in action.



## Why?

It's a quick way to edit and navigate code. It's especially useful if you want to edit multiple files at a time or take advantage of all the powerful code editing features of Visual Studio Code when making a quick change. For more information, see our [documentation](#).

> OUTLINE

> TIMELINE

GitHub Layout: U.S. ☒

<https://github.com/settings/billing/budgets>

Account budgets							New budget
⊕ Account Wytamma	100%	Product Packages	Stop usage Yes	\$0 spent	\$0 budget	...	
⊕ Account Wytamma	100%	Product Actions	Stop usage Yes	\$0 spent	\$0 budget	...	
⊕ Account Wytamma	100%	SKU All Premium Request SKUs	Stop usage Yes	\$0 spent	\$0 budget	...	
⊕ Account Wytamma	100%	Product Git LFS	Stop usage Yes	\$0 spent	\$0 budget	...	
⊕ Account Wytamma	100%	Product Codespaces	Stop usage Yes	\$0 spent	\$0 budget	...	