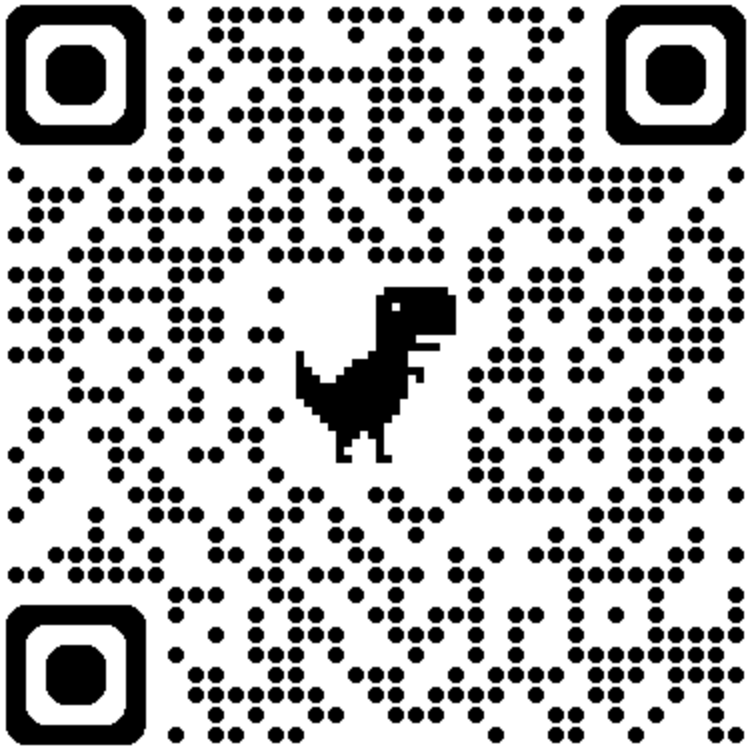
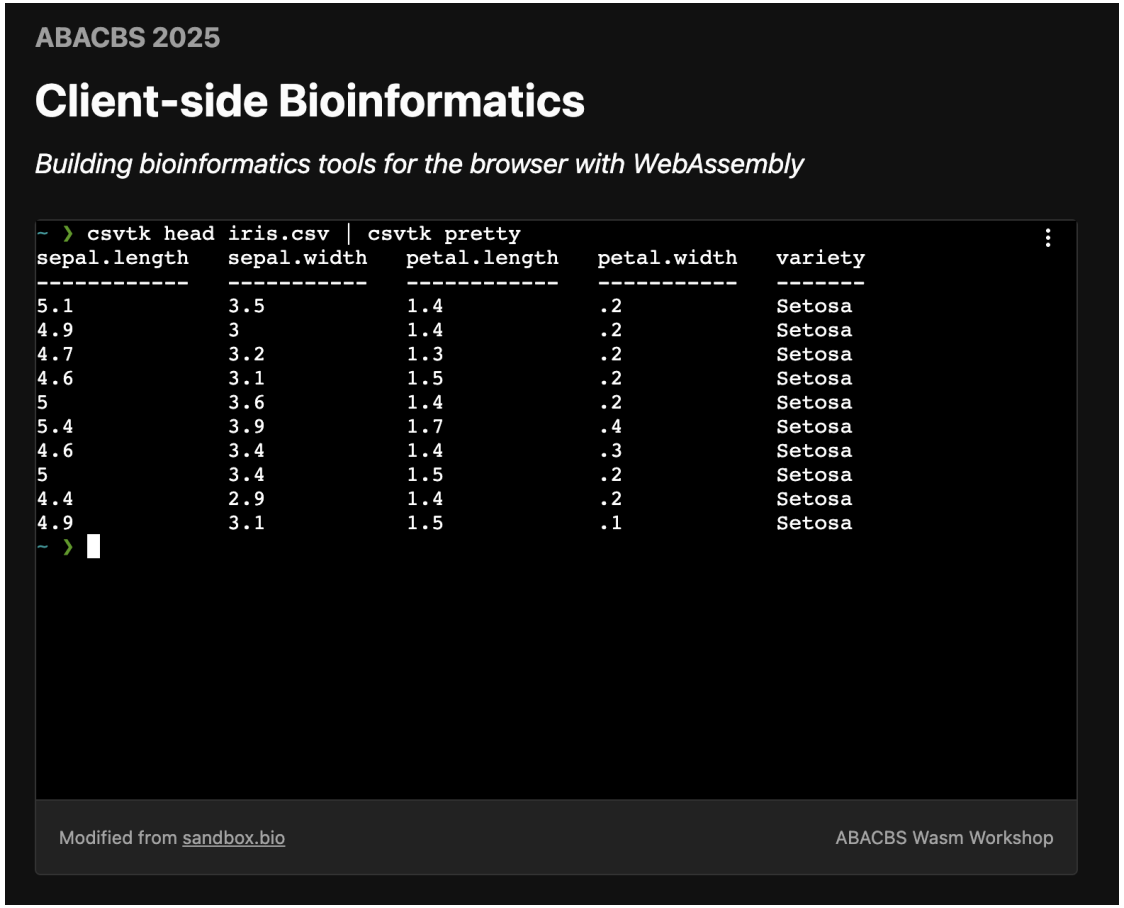
Client-side bioinformatics: building robust bioinformatics tools for the browser with WebAssembly

ABACBS 2025, Adelaide, 27-11-25

https://wasmodic.github.io/





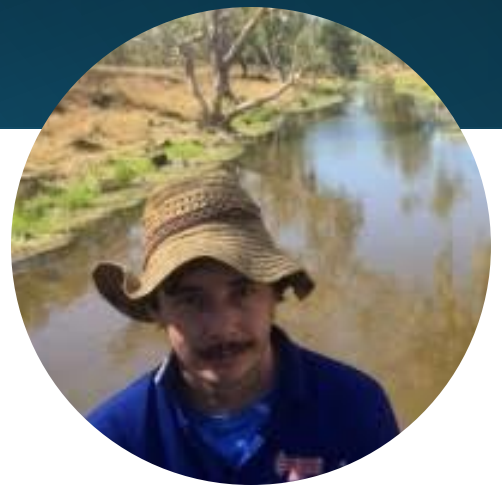
What is WebAssembly?

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

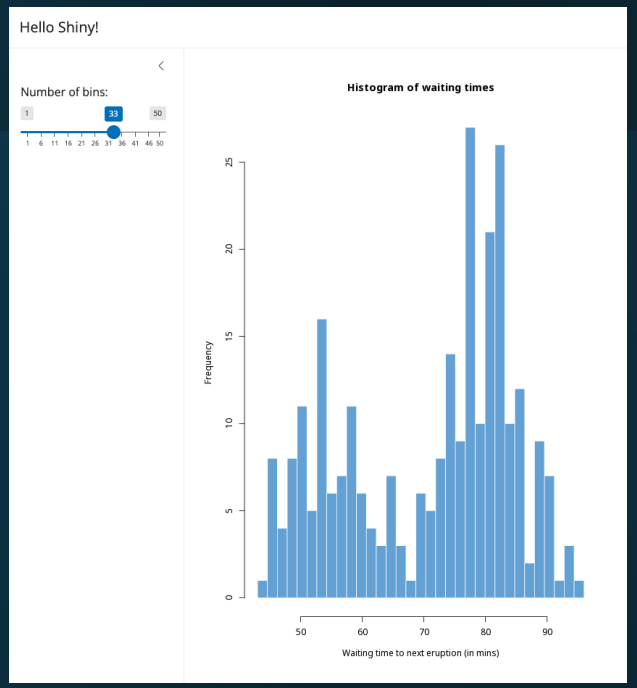
Make bioinformatics more accessible

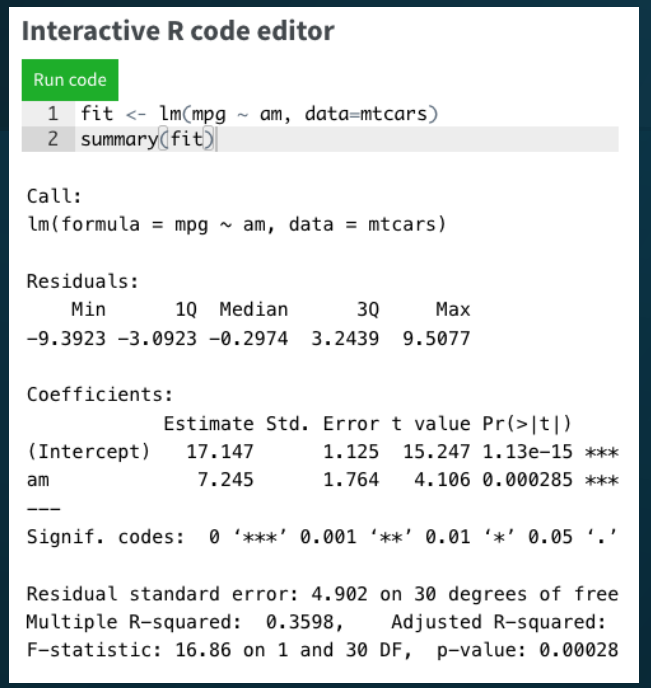
Increase the reach and impact of your tools

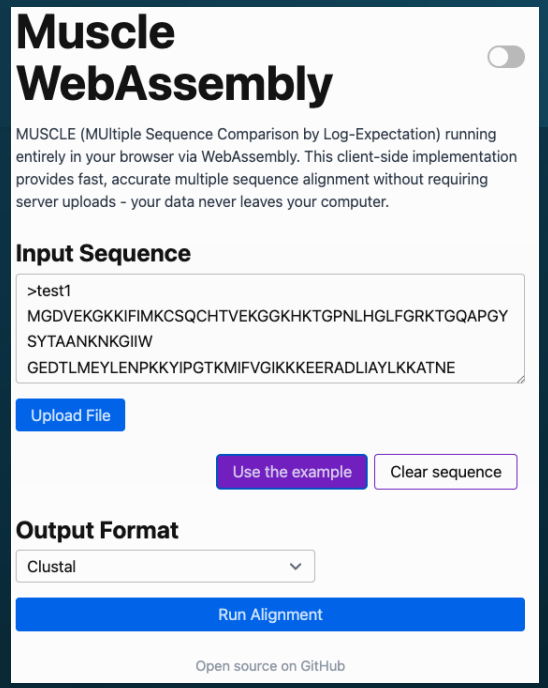
Improve reproducible science

Reduce code duplication

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







Who are you?

Mentimeter

Join at menti.com | use code 5625 1623

Mentimeter

Go to

www.menti.com

Enter the code

5625 1623

Or use QR code

← 💡 → Next slide

3

What is
Web Assembly ?

Torsten Seemann

THE UNIVERSITY OF MELBOURNE

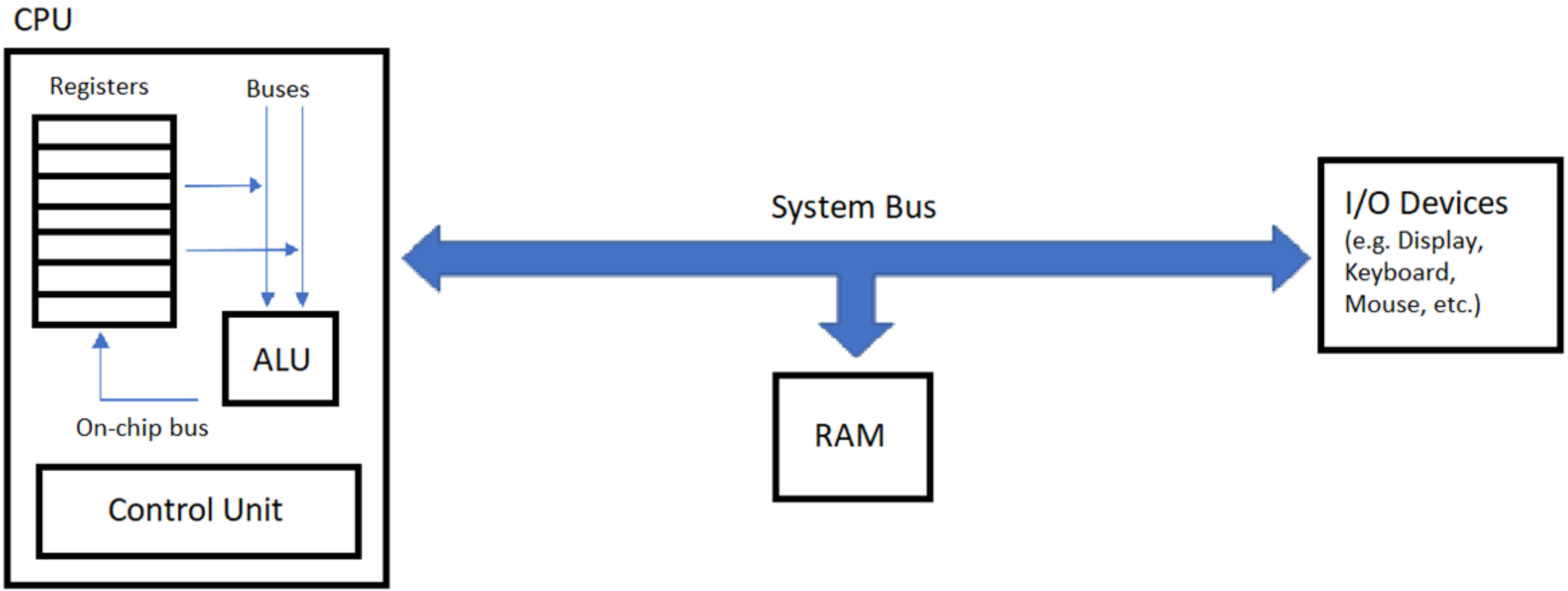
Doherty Institute

CENTRE FOR PATHOGEN GENOMICS

What is assembly?


Stations
of the Cross

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
```

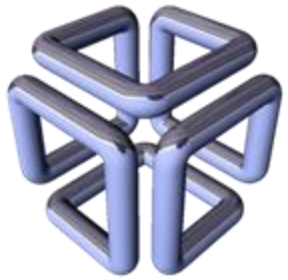
Compile .asm to machine ocde

```
.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)



Java Virtual Machine

Java byte code ~ CPU opcodes

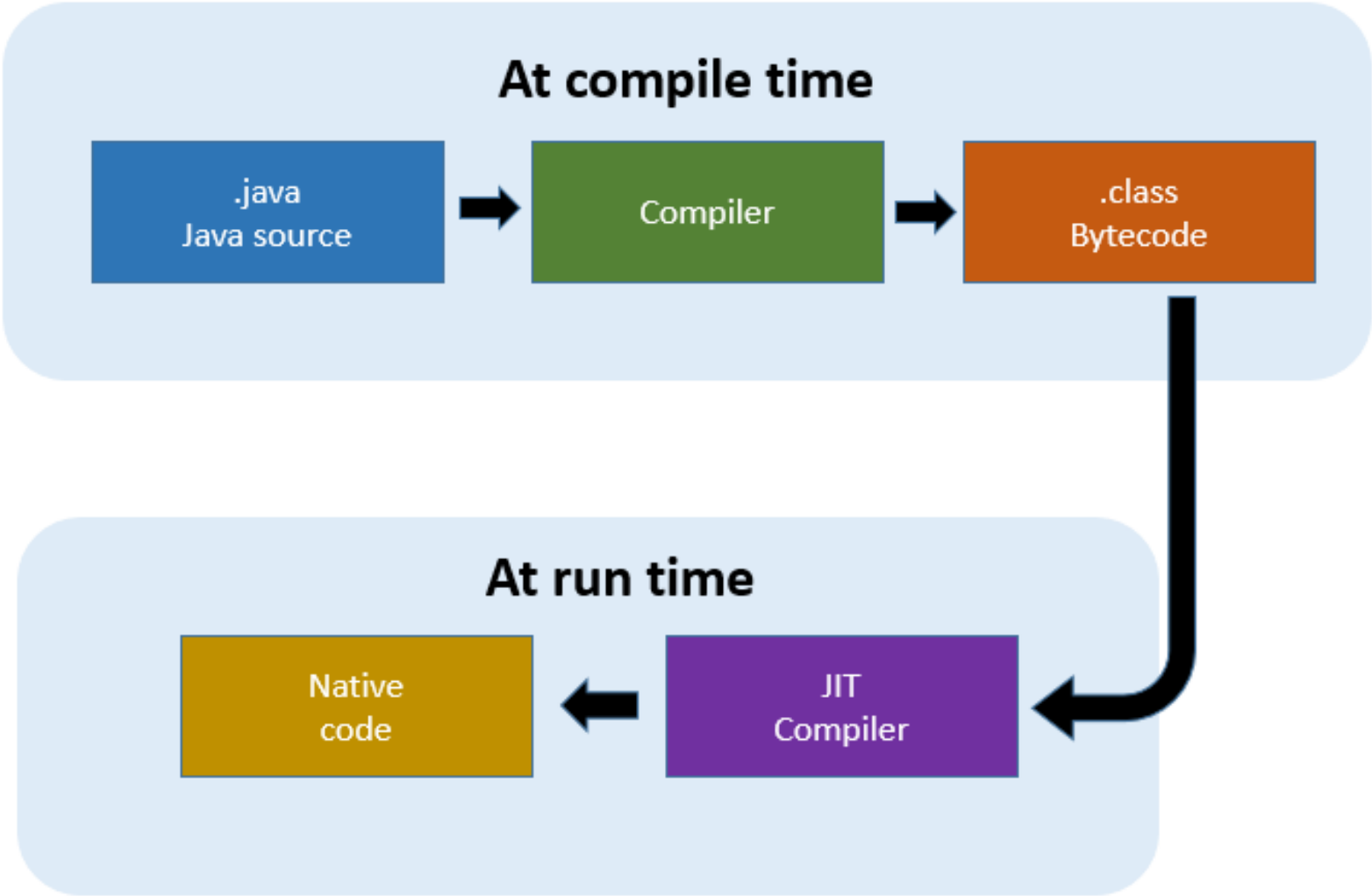
```
// Bytecode stream: 03 3b 84 00 01 1a 05
68 3b a7 ff f9
// Disassembly:

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

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.

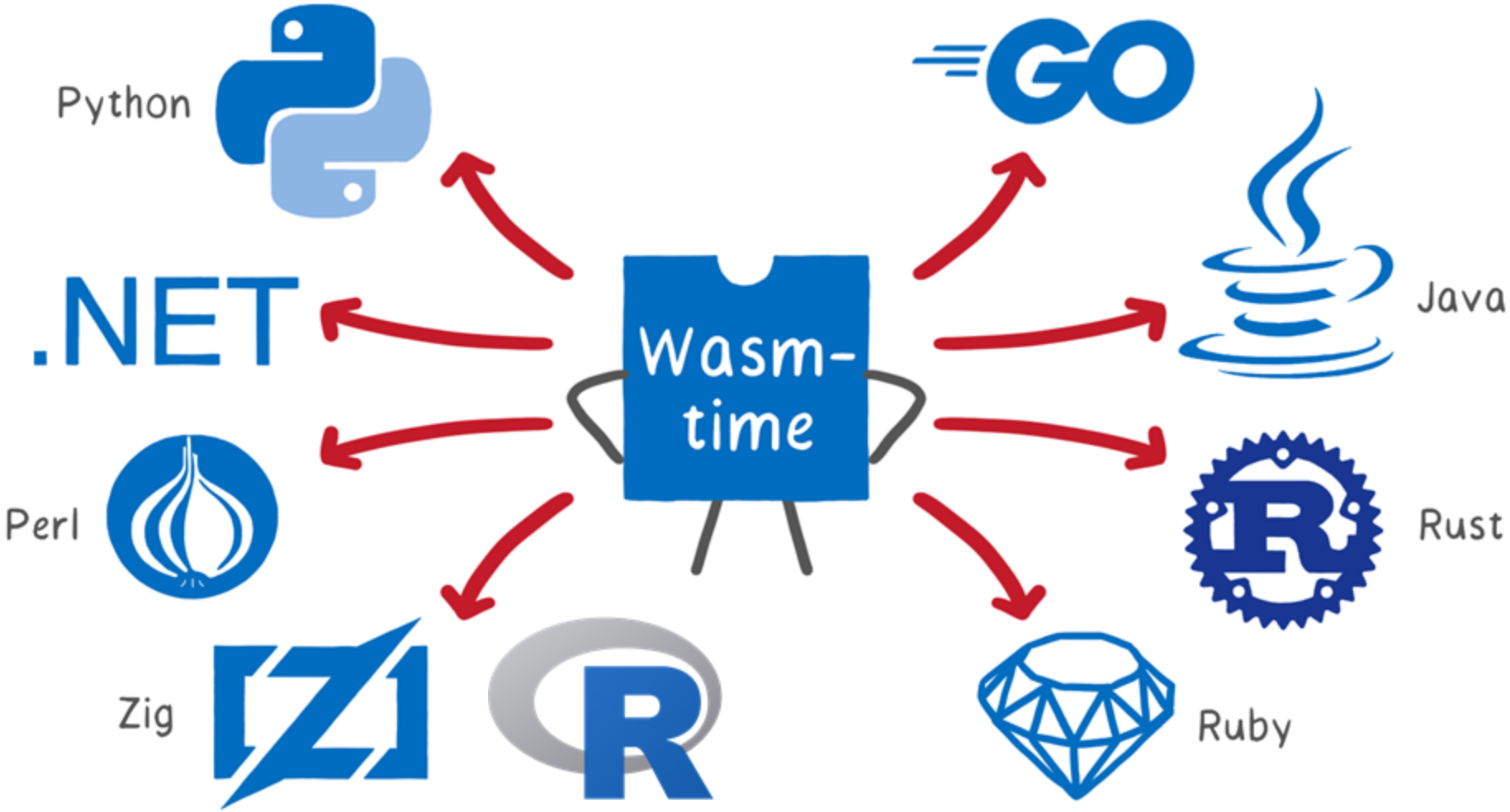
- “Stack based” architecture, strong typing
- Maps well to our register-based CPUs
- Supports use of SIMD and GPU

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

WASM = the new common CPU ?

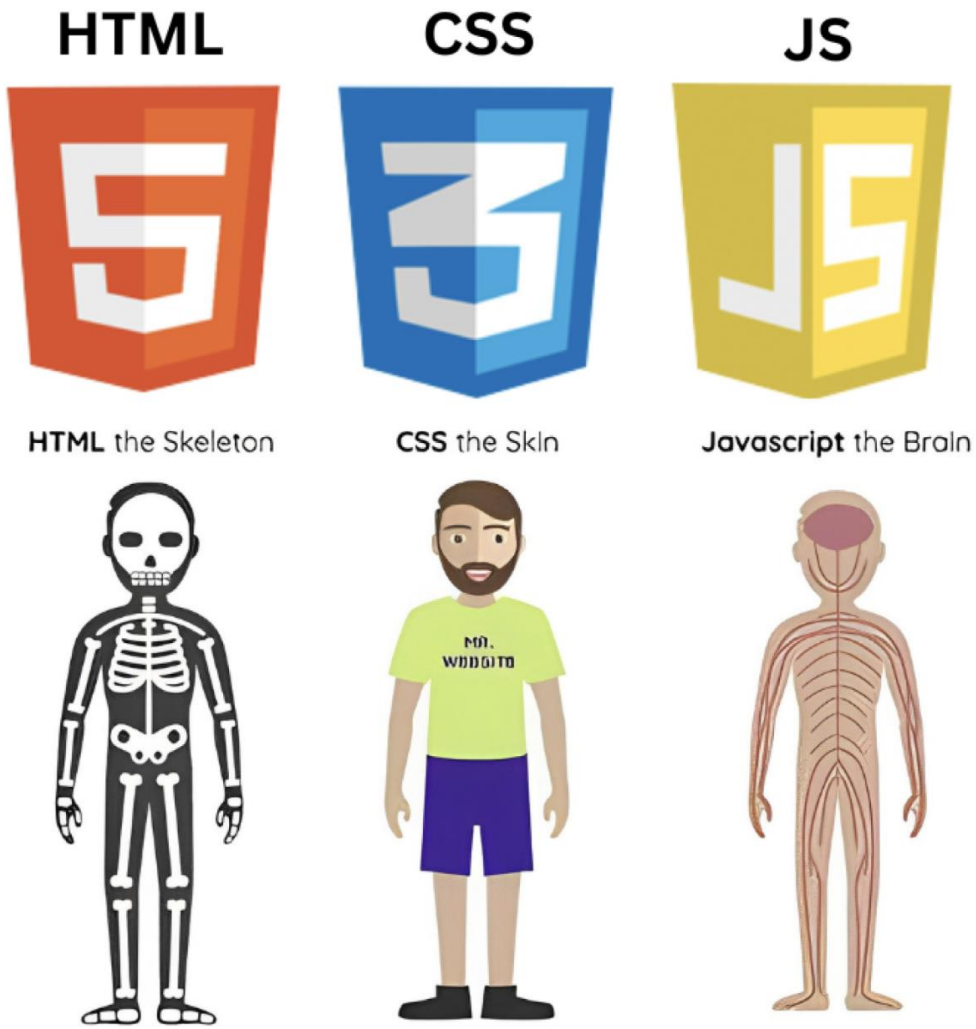


Thank you

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

JavaScript – CSS - HTML

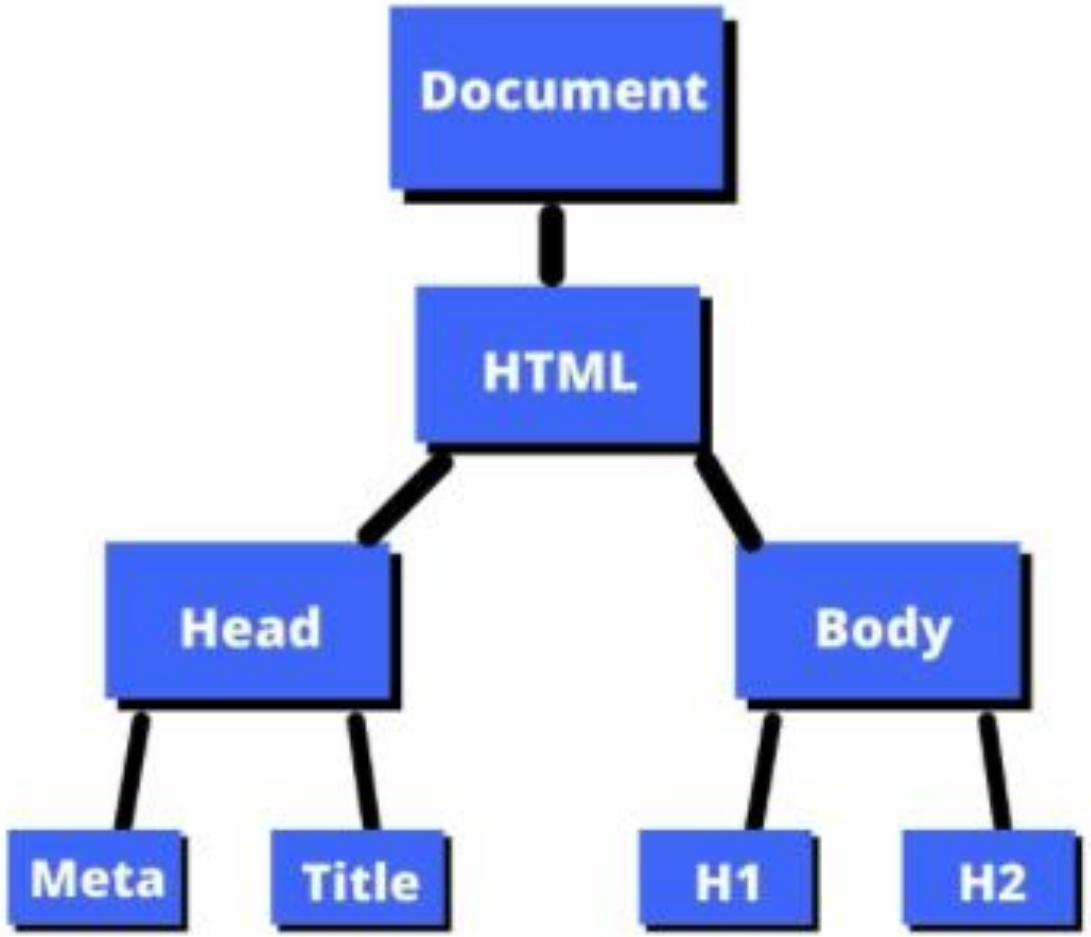


An analogy between HTML, CSS, and JavaScript

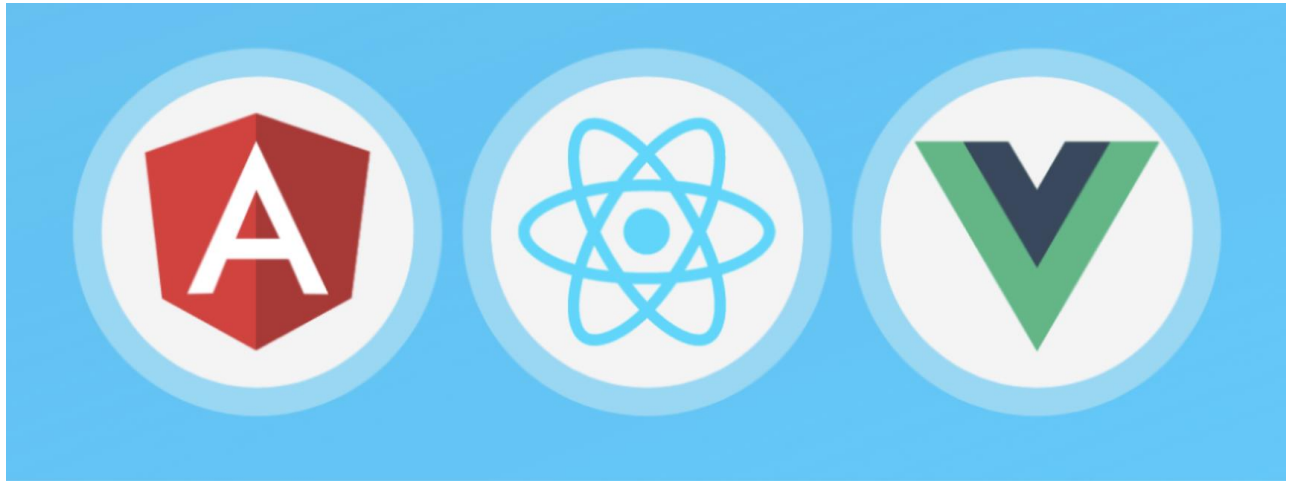
Figure taken from golden owl

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

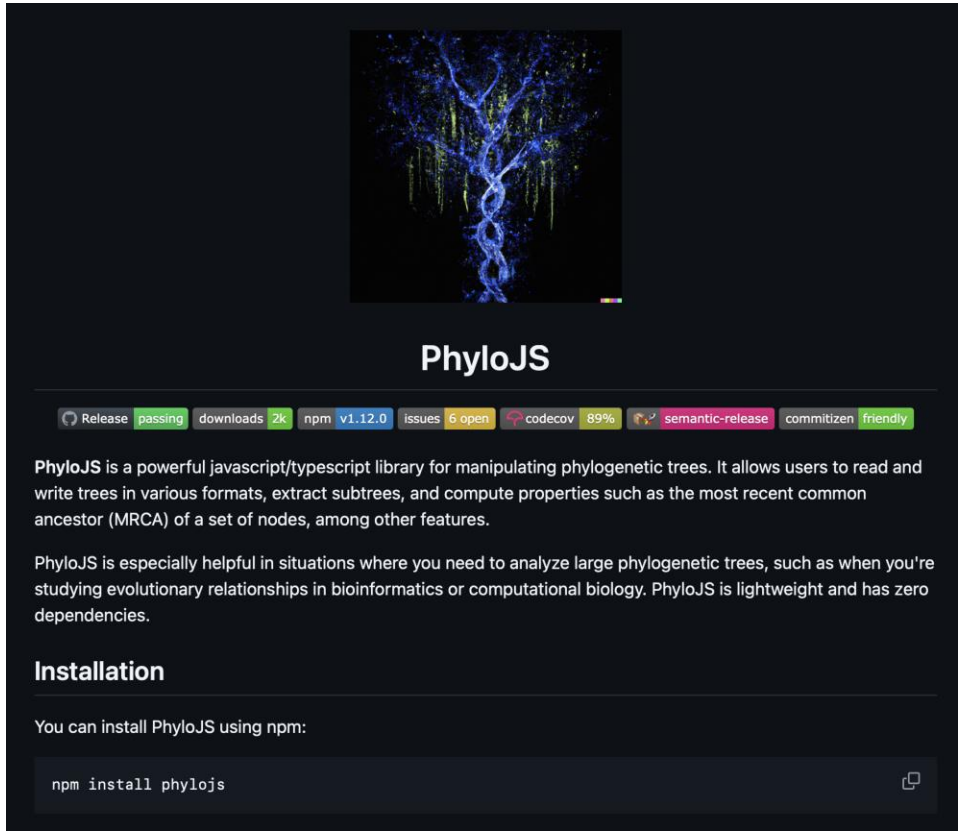
- Via "bindings" to JS

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 browsser into computers with near-native speed

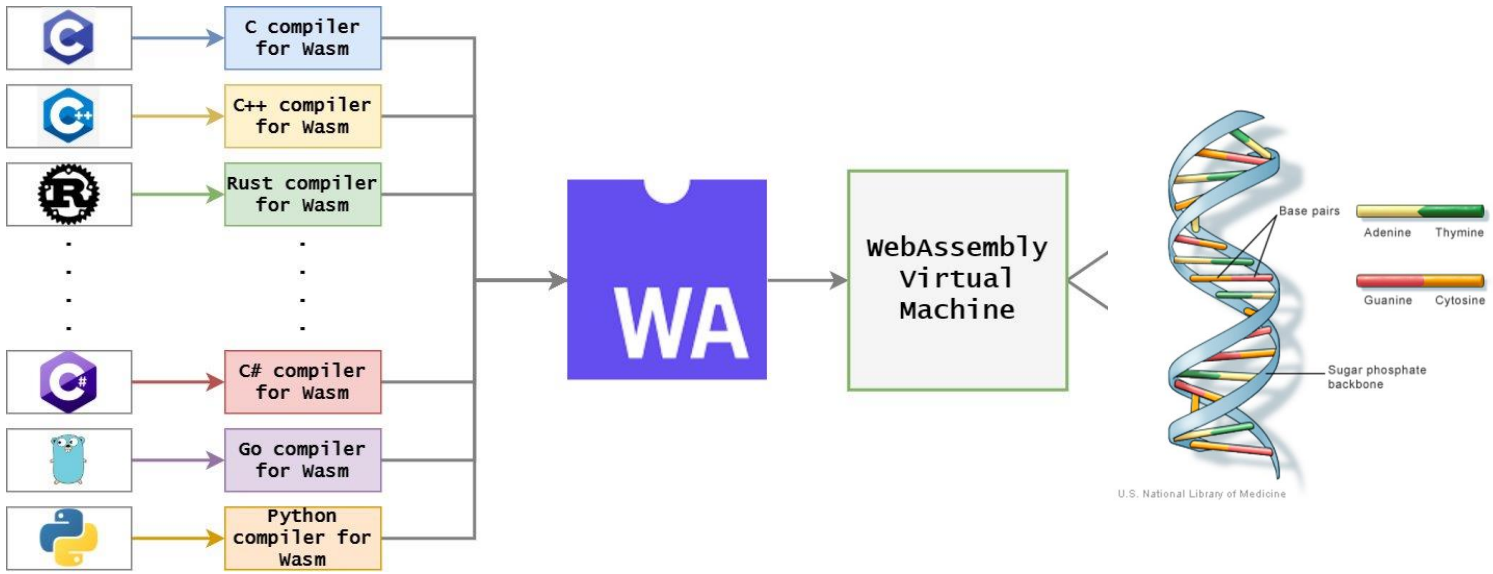
An example: if I had my time again



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

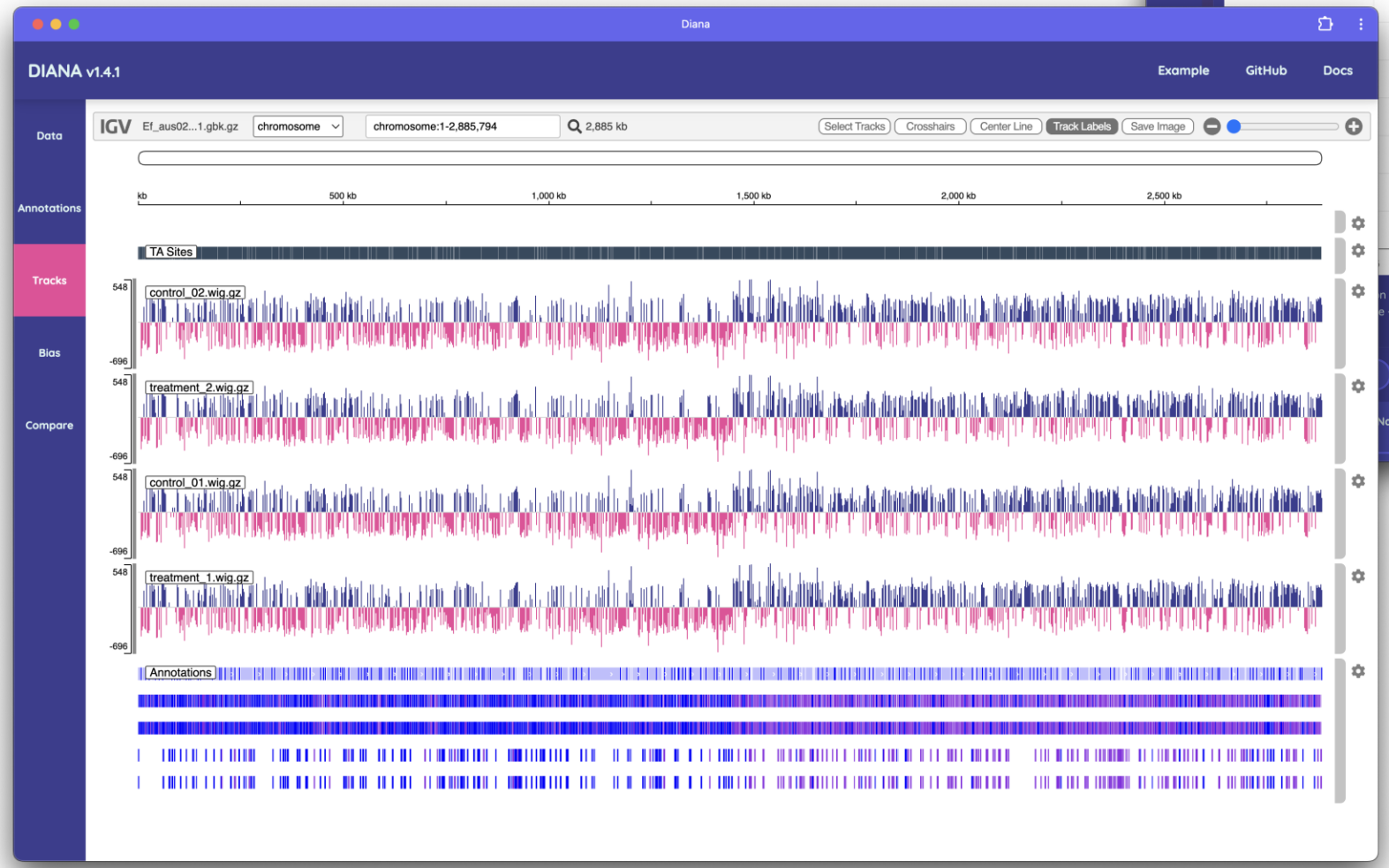
https://biowasm.com

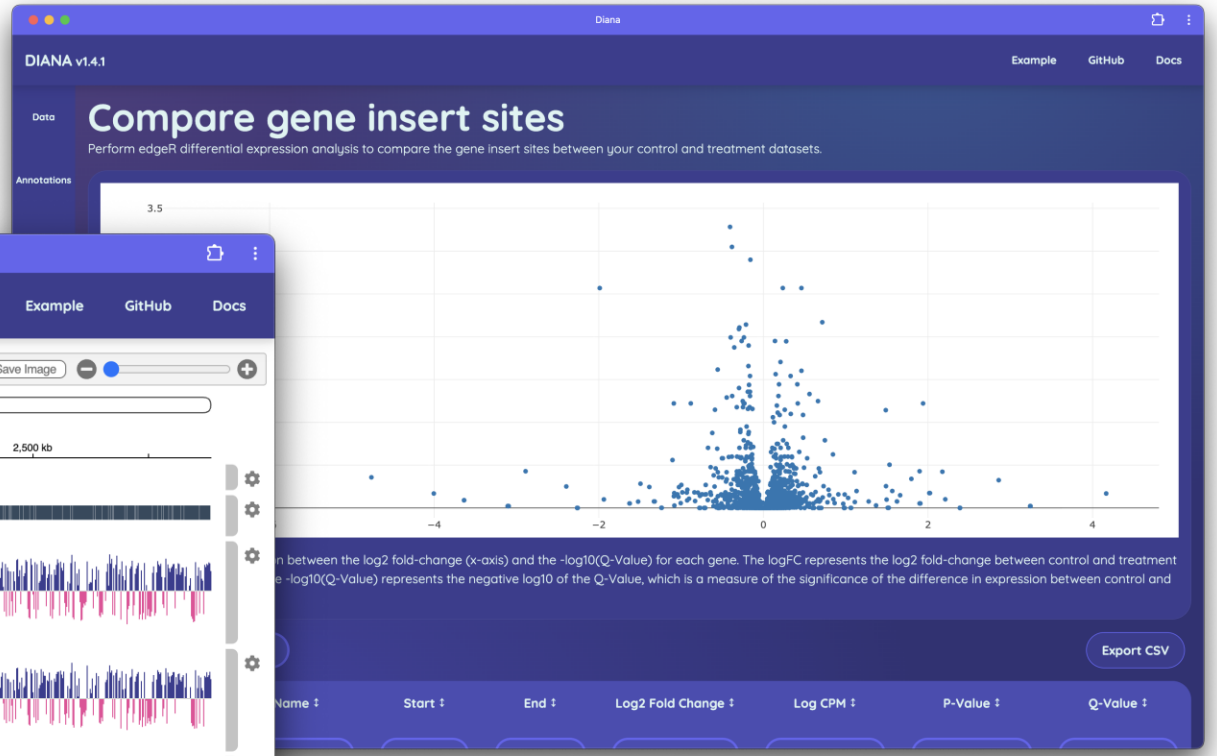
- **Run C/C++ genomics tools in the browser**
- Use bioinformatics command-line tools in your web apps with just a few lines of code. Powered by WebAssembly.

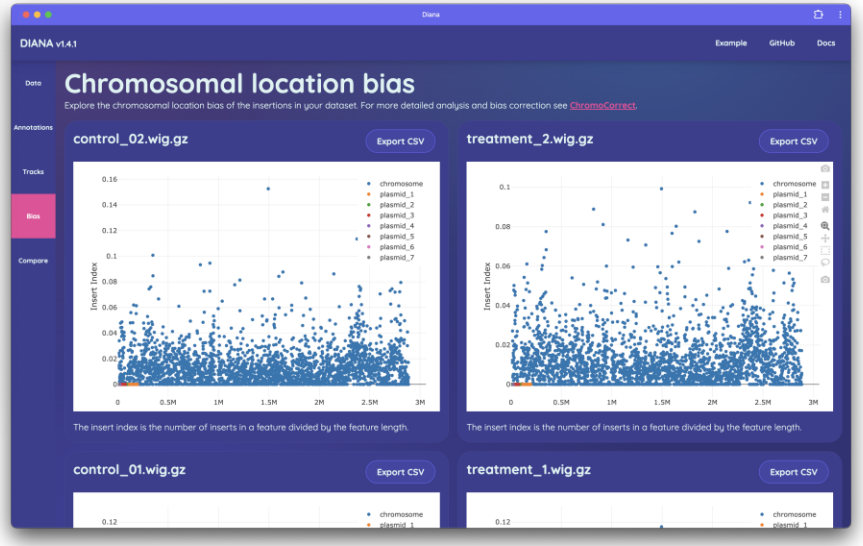
https://sandbox.bio

- Interactive bioinformatics tutorials. No setup required.

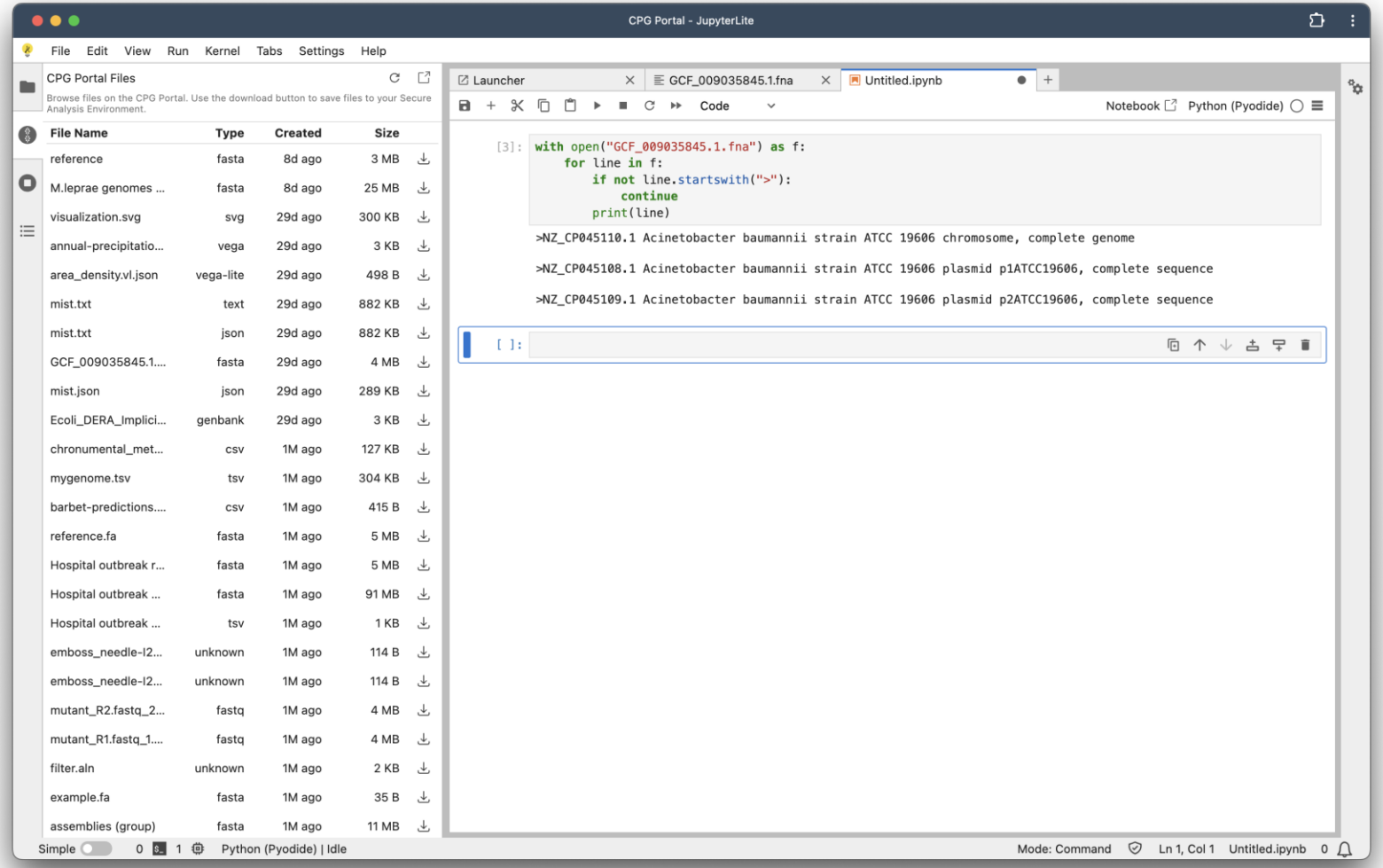
https://diana.cpg.org.au/





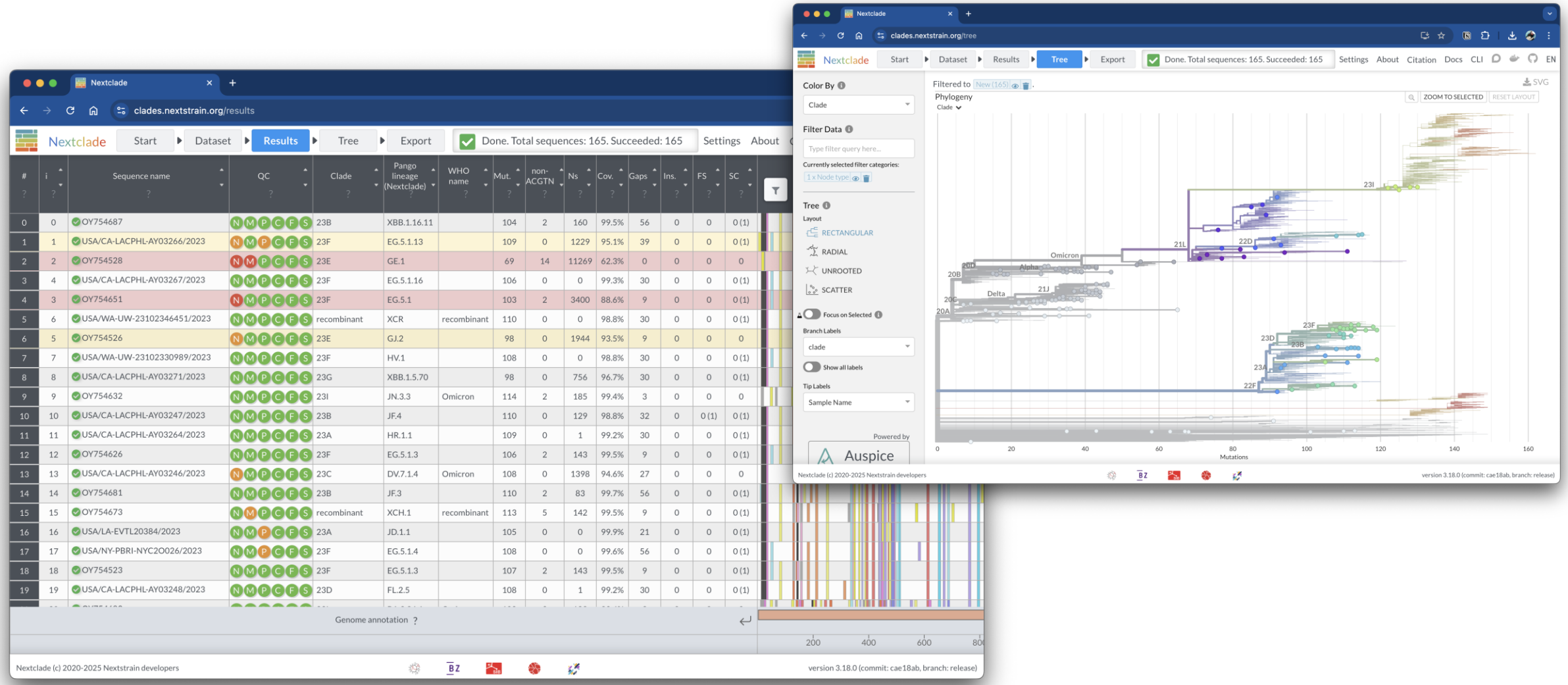


https://portal.cpg.unimelb.edu.au/



CENTRE FOR PATHOGEN GENOMICS

https://clades.nextstrain.org/



Killer Features

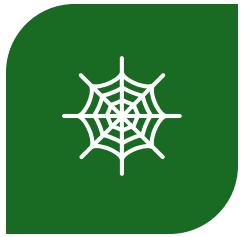
WebAssembly is not a silver bullet, but it opens a new niche for software development i.e., real software in the browser.

Reasons why people are excited about WASM:

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

WebAssembly will not solve everything - There are many limitations

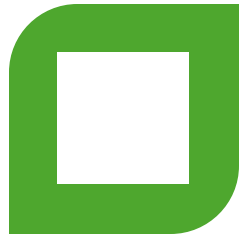
CLIENT-SIDE IS INHERENTLY OPEN-SOURCE (NO SECRETS)



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

Check for updates

RESEARCH ARTICLE

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

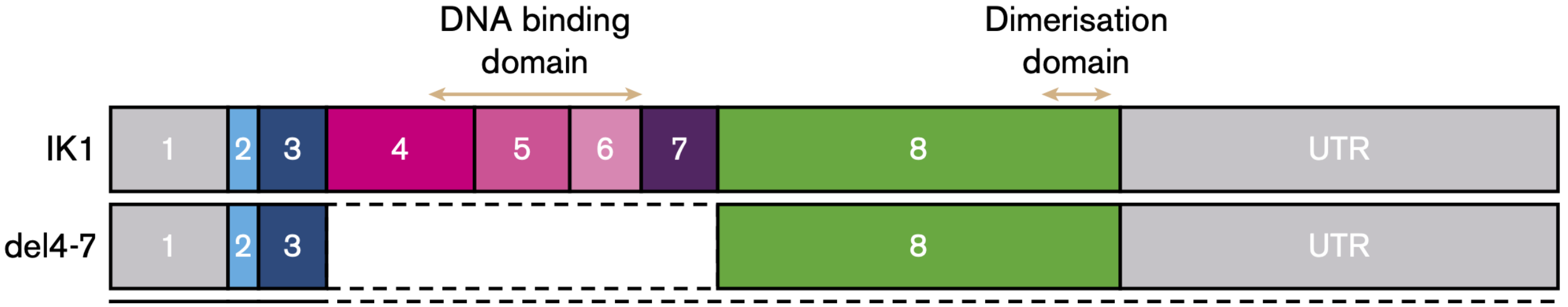
[version 1; peer review: 2 approved]

Andrew Lonsdale [1-3], Andreas Halman[1,3], Lauren Brown[2,4,5], Hansen Kosasih [2], Paul Ekert[1-5], Alicia Oshlack [1,3,6]

[1]Peter MacCallum Cancer Centre, Parkville, VIC, 3052, Australia
[2]Murdoch Children's Research Institute, Parkville, VIC, 3052, Australia
[3]Sir Peter MacCallum Department of Oncology, University of Melbourne, Parkville, VIC, 3010, Australia
[4]Children's Cancer Institute Australia, Sydney, NSW, 2052, Australia
[5]School of Women's and Children's Health, UNSW Sydney, Sydney, NSW, 2052, Australia
[6]School of Mathematics and Statistics, University of Melbourne, Parkville, VIC, 3010, Australia

Toblerone

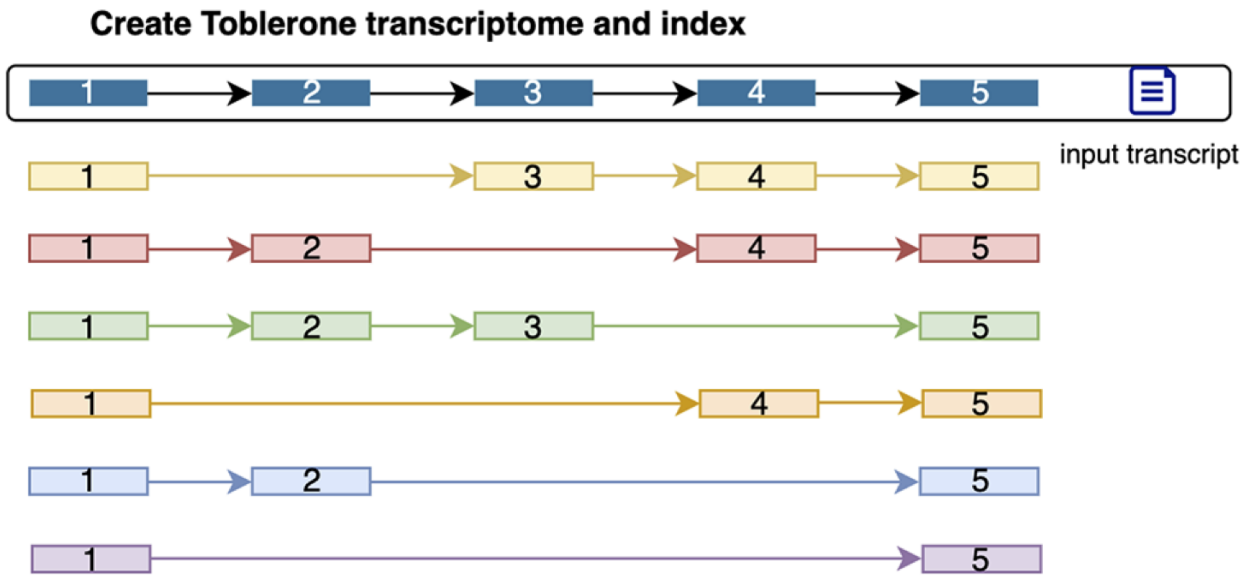
Toblerone is a RNA-seq **pseudoalignment** approach that uses a reference of **only exon deletion transcripts** and counts reads unique to those transcripts to identify exon deletions in cancer.



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

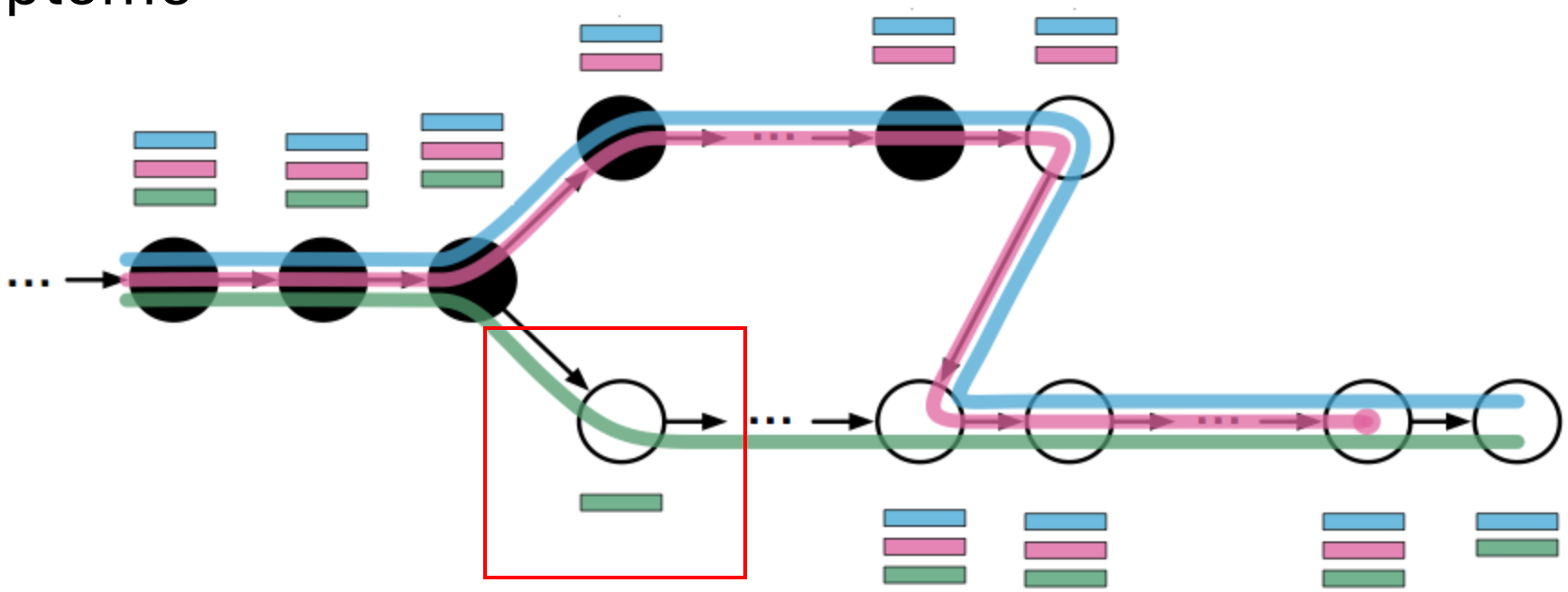
Toblerone

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



Simplified pseudoalignment

Because of the way the transcriptome is consctucted, 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 expresion

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

Mapping

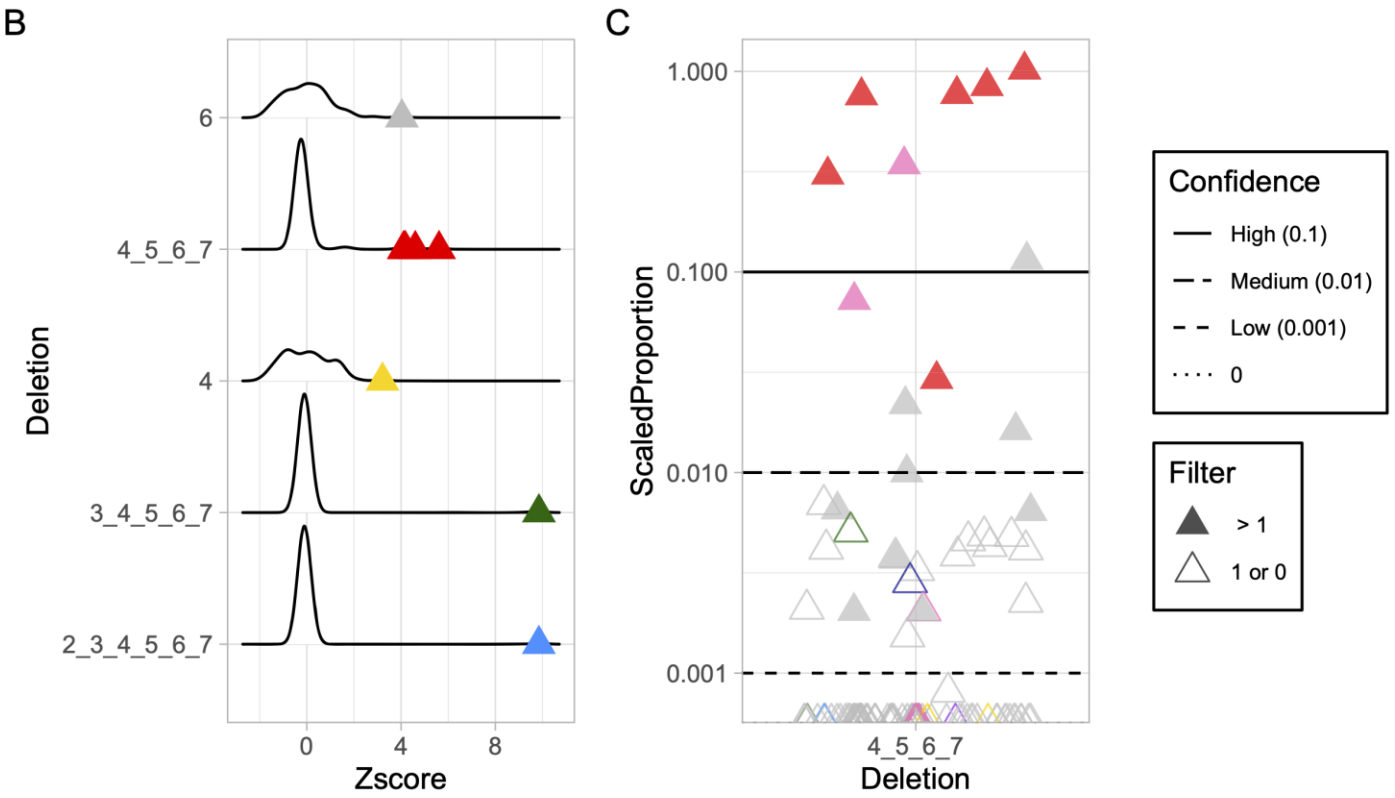
- Given the de Bruijn index for a gene, e.g. IKZF1, reads are mapped and the scaled proportions for each of the possible exon deletions are calculated

- 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

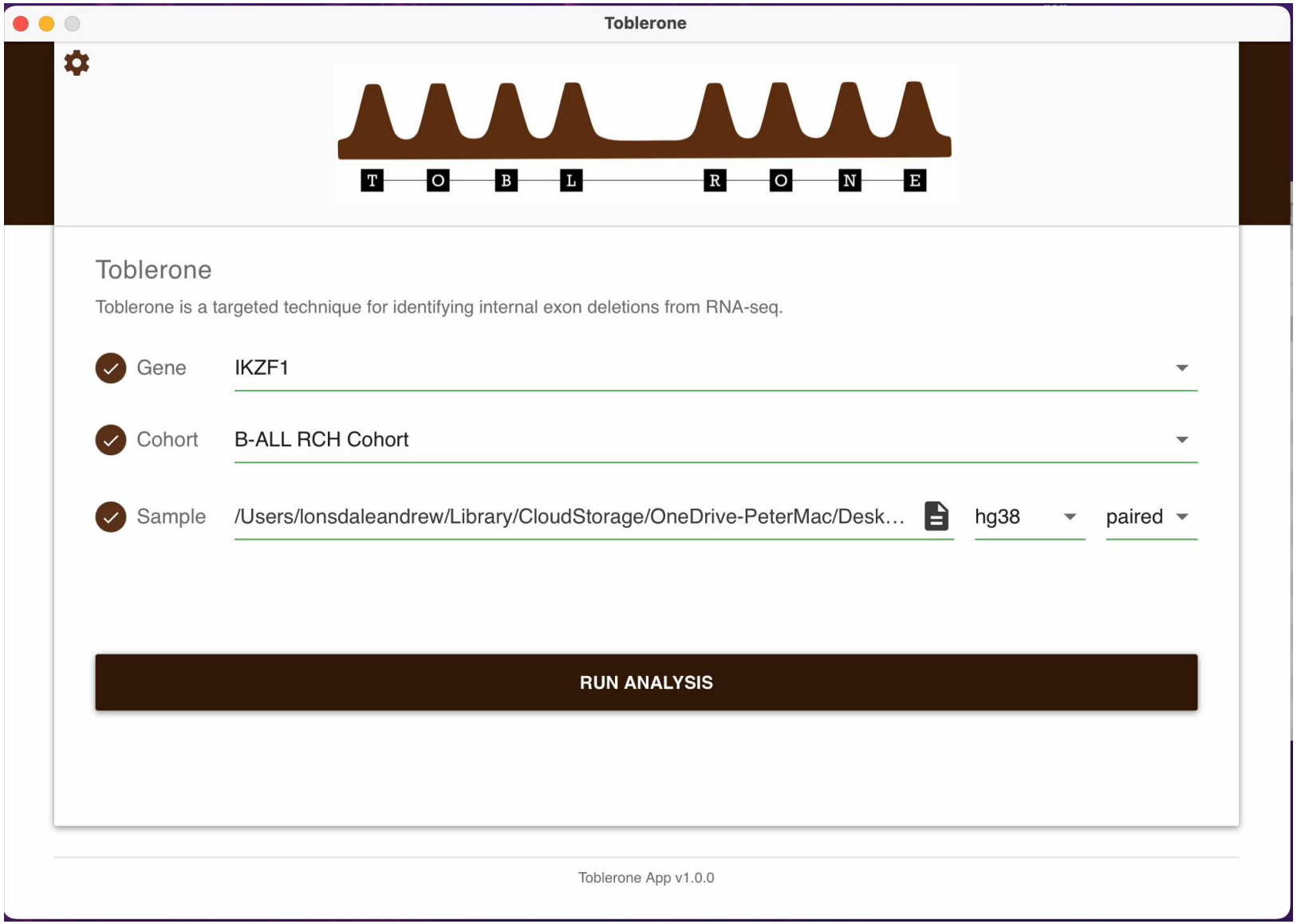
- Custom pseudoaligner (tinyt)
- Command line tool – **index** and **map**
- FASTQ input
- Generic approach
- Apply to many samples in parallel
- Output CSV

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

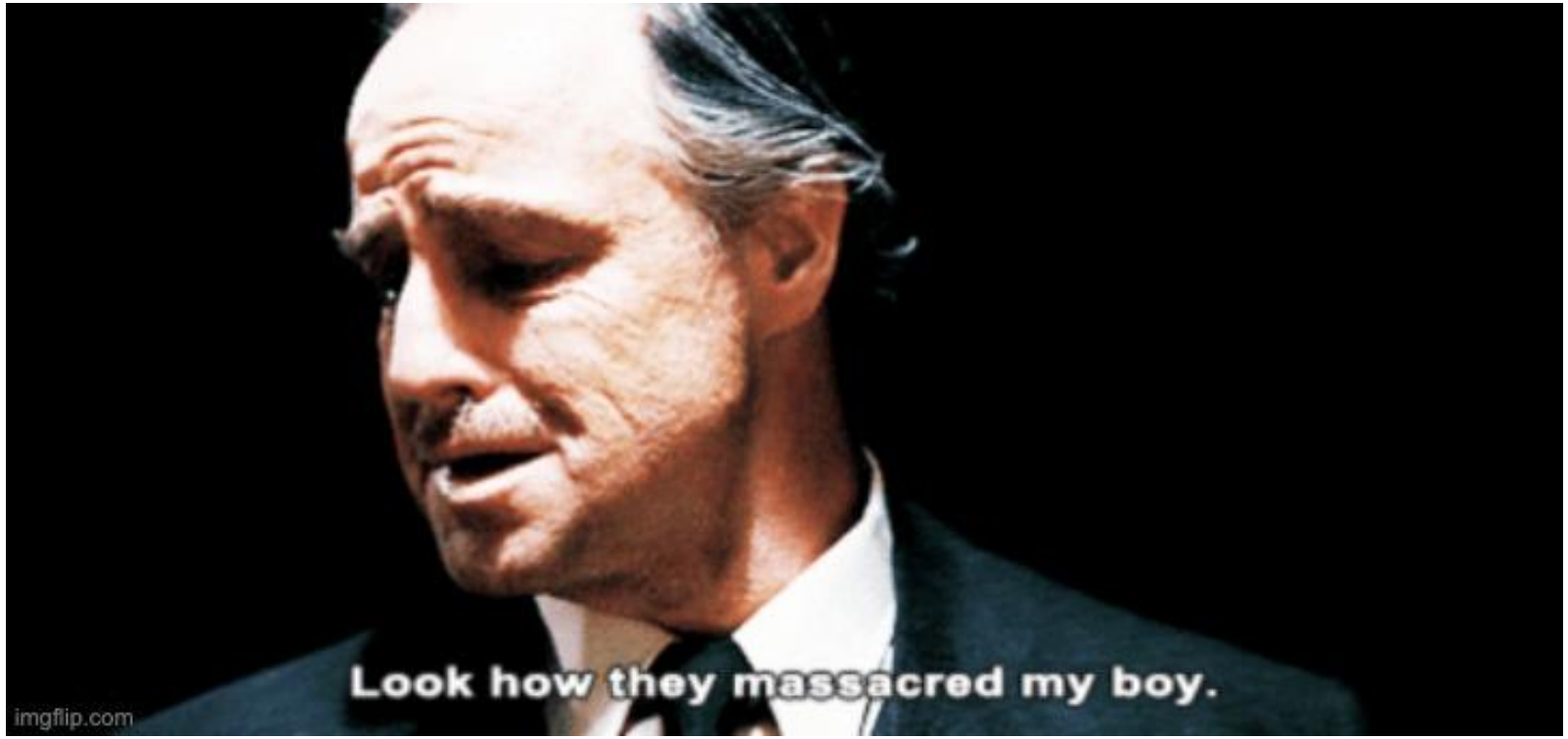
Compromises

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



Compromises

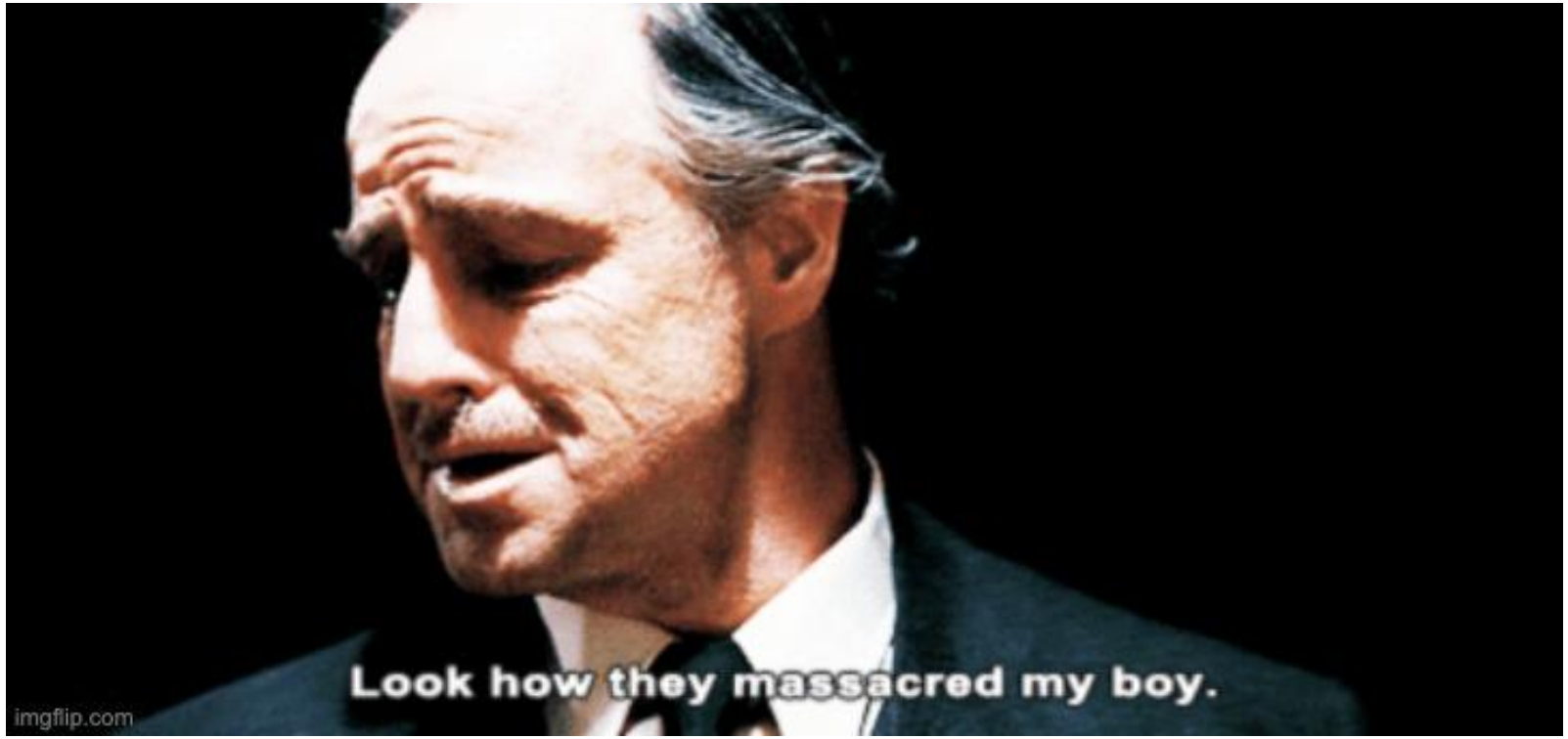
- Lost ggplot
- New JS plotting with HighCharts library
- Operating system install issues, especially on WSL2
- Custom code to extract reads from BAM files



Compromises

- Successful…but it never entirely satisfied

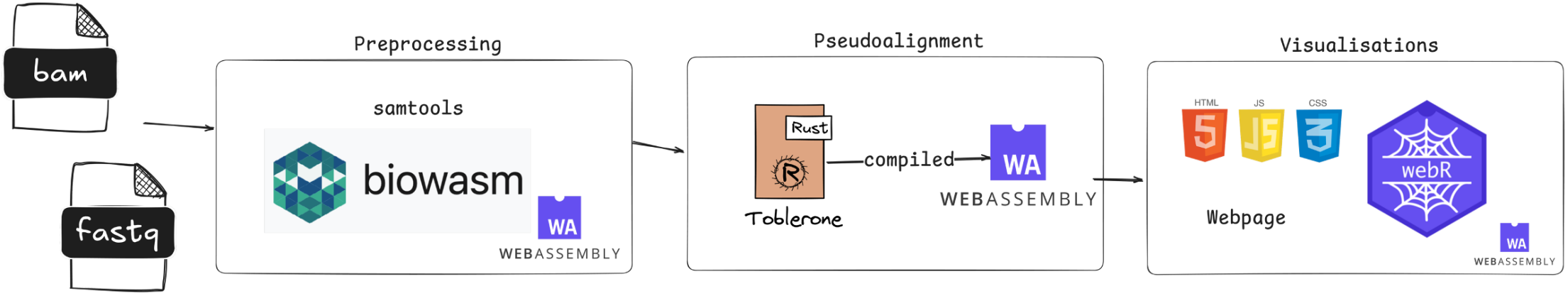
- Published though, so time to move on…



Web Assembly

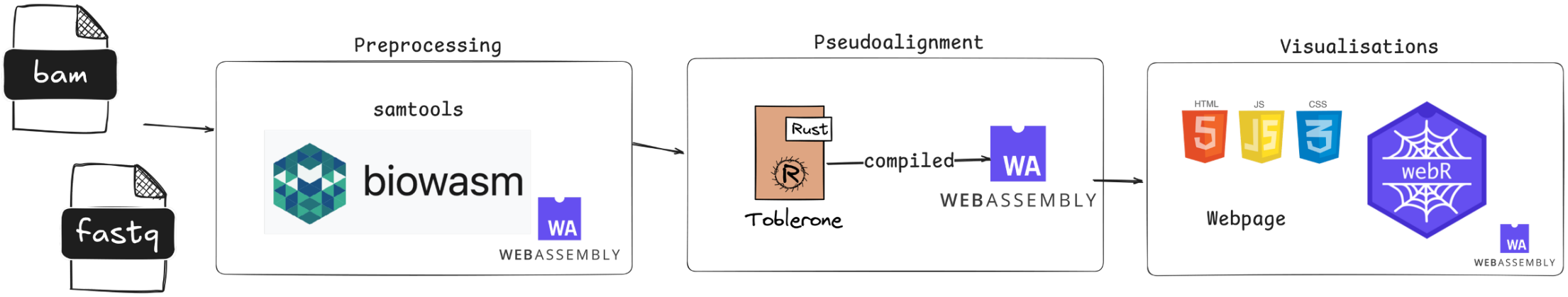
Toblerone WASM

- 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 Pythons 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

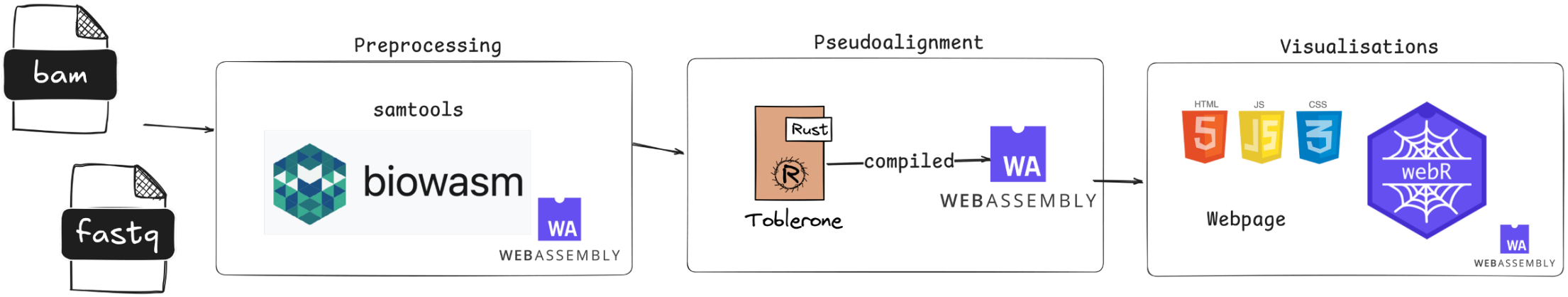


Toblerone WASM

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



Toblerone WASM

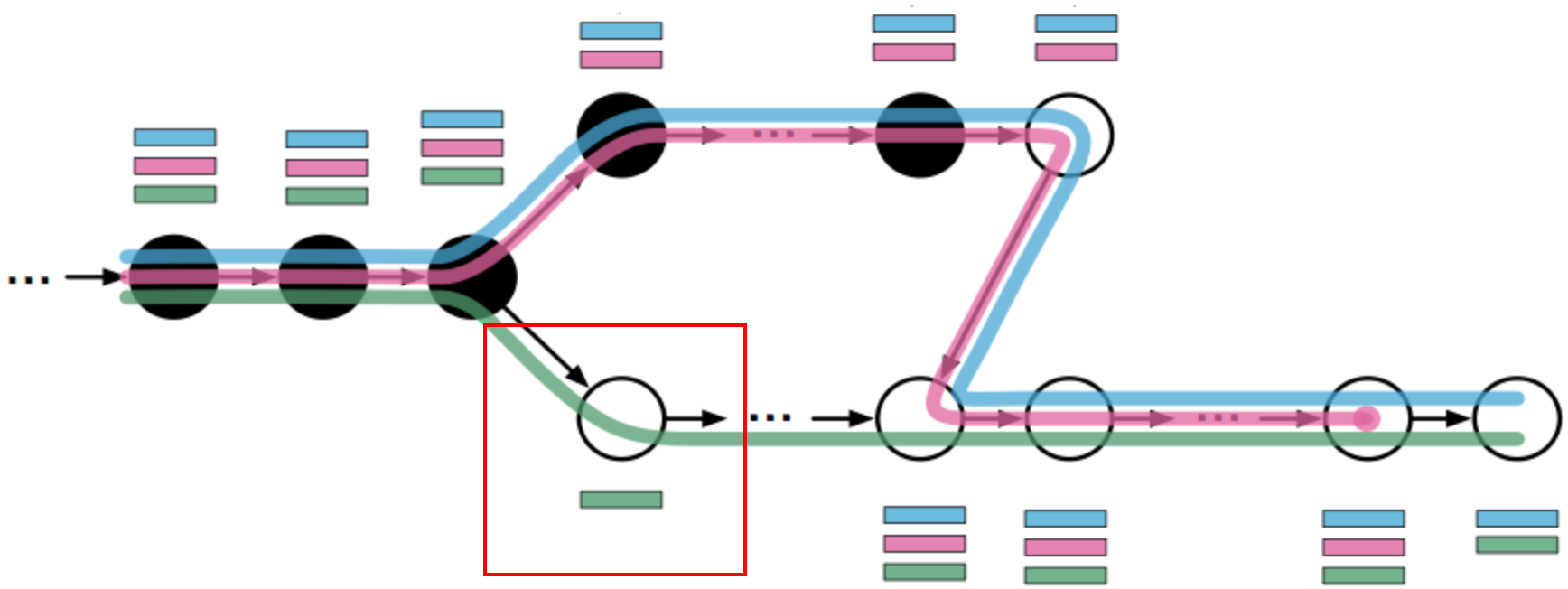


Conversion lessons

- 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 platfrom won't work direclty
  - 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 essentialy 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 essentialy 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 Equivalance 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 +          };
```

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};
```

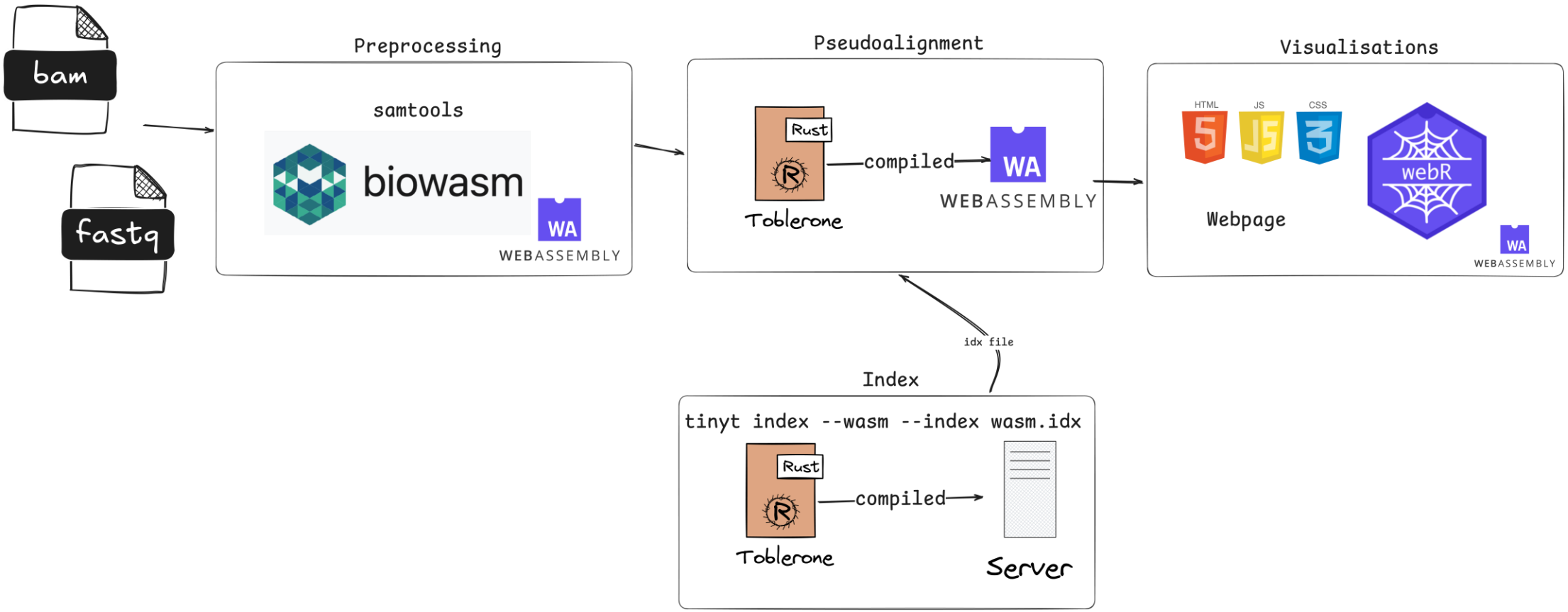
```
133 + pub trait IndexLike: Sync {
134 +     fn map_read(&self, read_seq: &DnaString, mismatch_size: usize) -> Option<(Vec<u32>, usize, usize, usize)>;
135 +     fn tx_names(&self) -> &Vec<String>;
136 +     fn tx_gene_mapping(&self) -> &HashMap<String, String>;
137 +     fn gene_length_mapping(&self)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

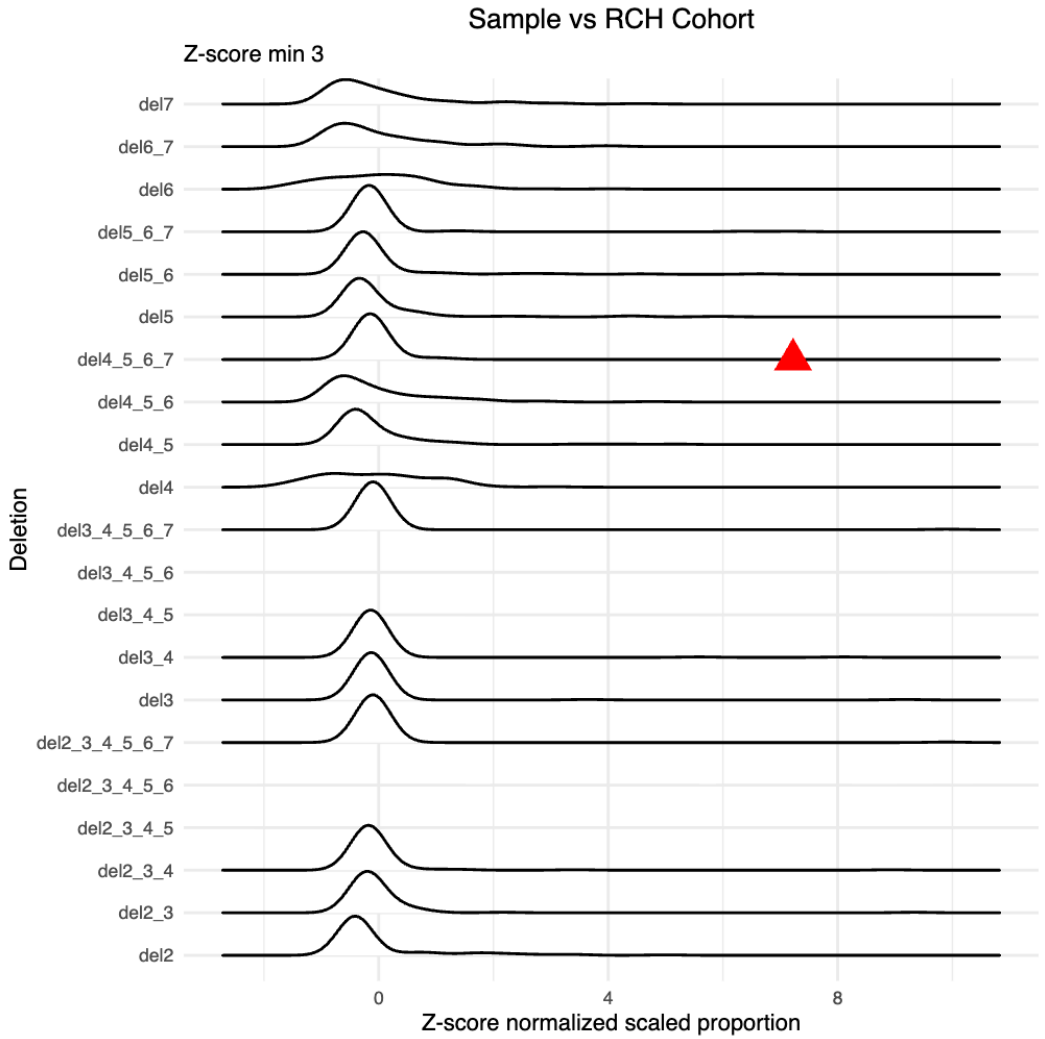
IndexLike

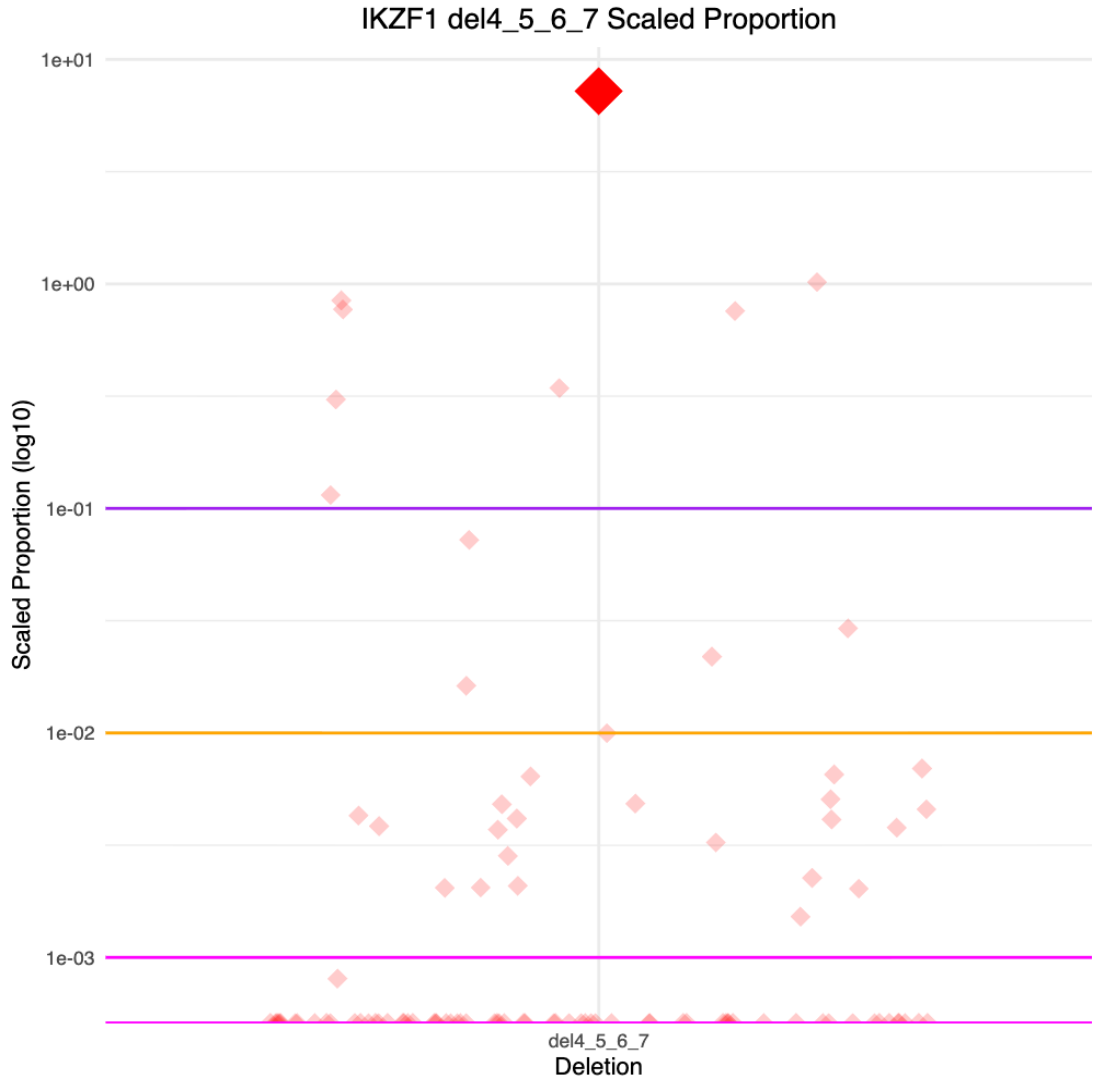
```
345 + impl IndexLike for WasmRuntimeIndex {
346 +     fn map_read(&self, read_seq: &DnaString, _allowed_mismatches: usize) -> Option<(Vec<u32>, usize, usize, usize)> {
347 +         // convert read into canonical k-mer u64s (utils::kmers_to_u64_vec must match exporter encoding)
348 +         let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

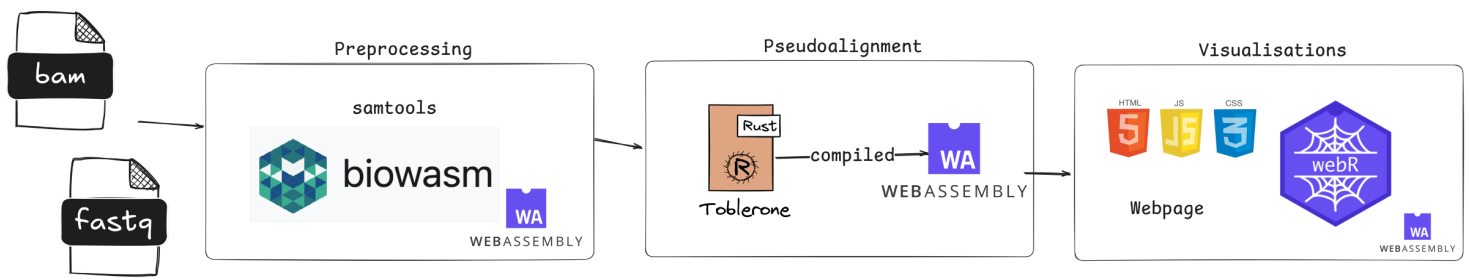
```
415 + // Exporter: walk the de Bruijn graph and collect all kmers -> node+offset
416 + pub fn export_wasm_index<K: Kmer>(al: &Pseudoaligner<K>) -> WasmIndex {
417 +     let mut kmers = Vec::new();
418 +     let klen = K::k() as u8;
```

Toblerone WASM




Sample vs RCH Cohort
Z-score min 3
del7
del6_7
del6
del5_6_7
del5_6
del5
del4_5_6_7
del4_5_6
del4_5
del4
del3_4_5_6_7
del3_4_5_6
del3_4_5
del3_4
del3
del2_3_4_5_6_7
del2_3_4_5_6
del2_3_4_5
del2_3_4
del2_3
del2
Deletion
0
4
8
Z-score normalized scaled proportion


IKZF1 del4_5_6_7 Scaled Proportion
1e+01
1e+00
1e-01
1e-02
1e-03
Scaled Proportion (log10)
del4_5_6_7
Deletion


bam
fastq
Preprocessing
samtools
biowasm
WEBASSEMBLY
Pseudoalignment
Rust
Toblerone
compiled
WA
WEBASSEMBLY
Visualisations
HTML
JS
CSS
webR
Webpage
WA
WEBASSEMBLY

IKZF1 – WebR ggplot PNG demo

Elements Console Sources Network Performance Memory 1

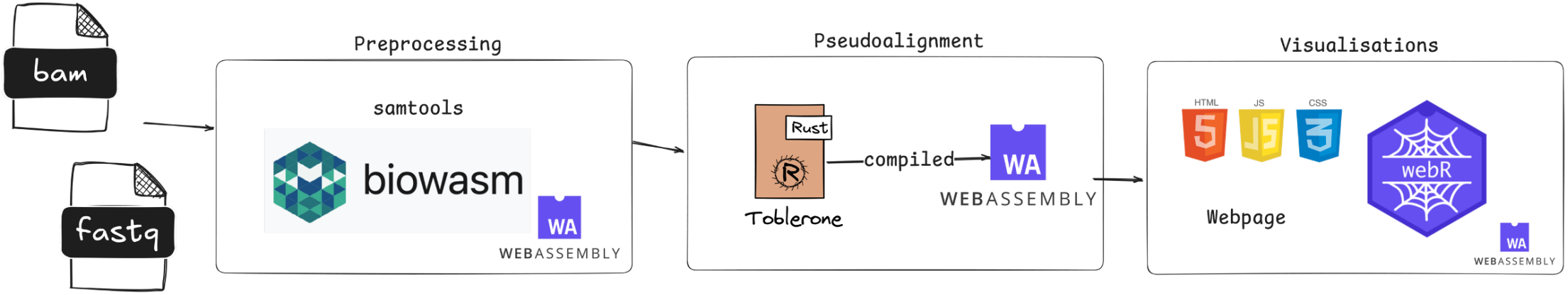
top Filter All levels No Issues

Refused to get unsafe header "Content-Encoding" R.js:1822

Console What's new Issues

Summary

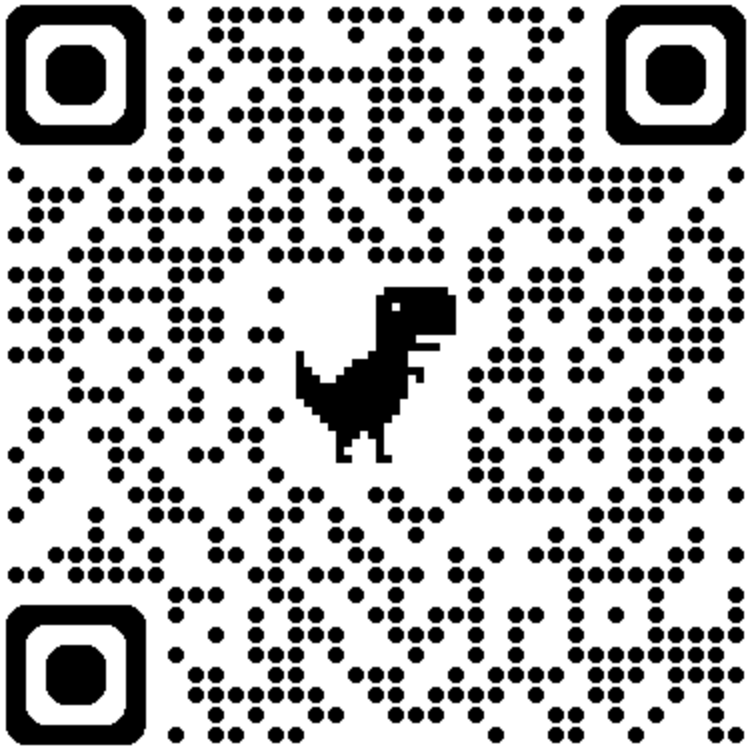
- Using WebAssembly (WASM), we created a client-side web browser version of Toblerone that reduces maintenance effort, and enhances accessibility and privacy.
- It is under development and available at https://github.com/Oshlack/TobleroneWASM

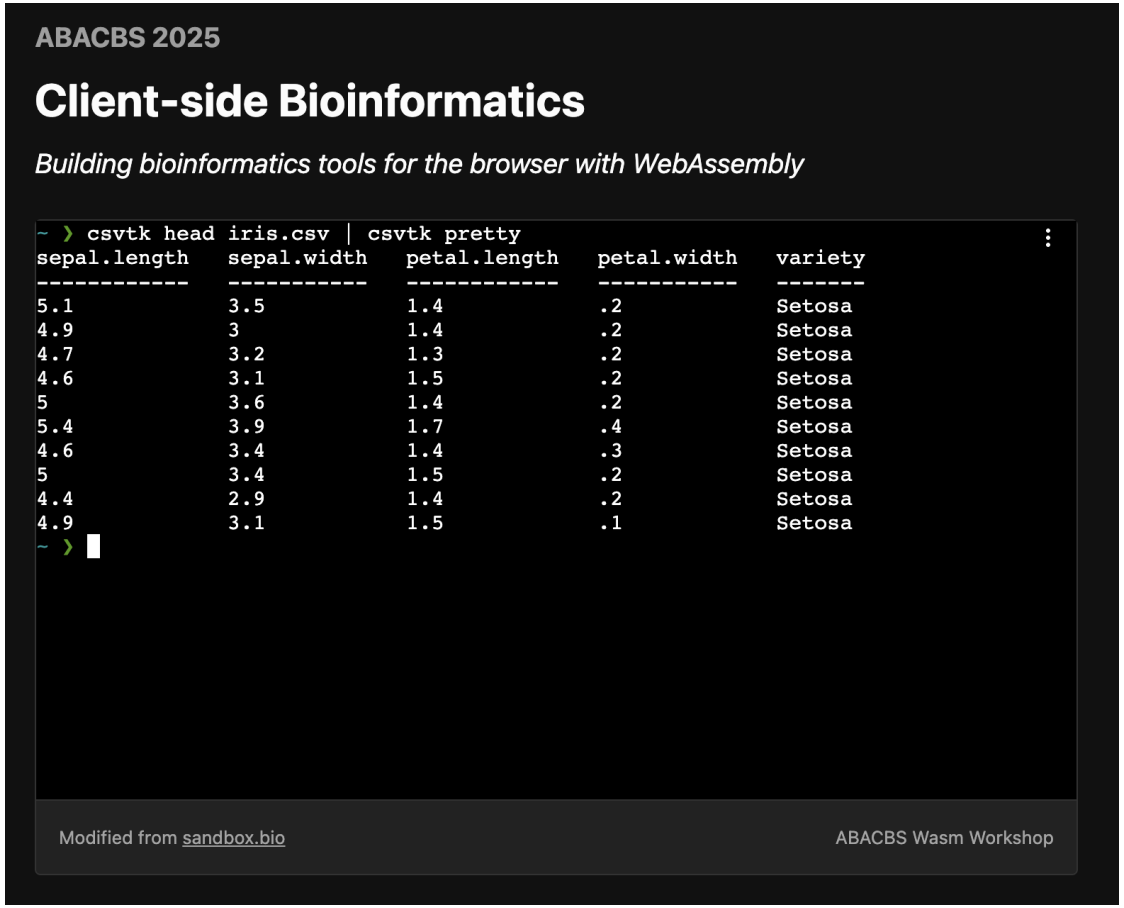


Conclusion

- 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 ToblerooneApp 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 unversal feature, what might not be avilable in WASM version of tool

https://wasmodic.github.io/

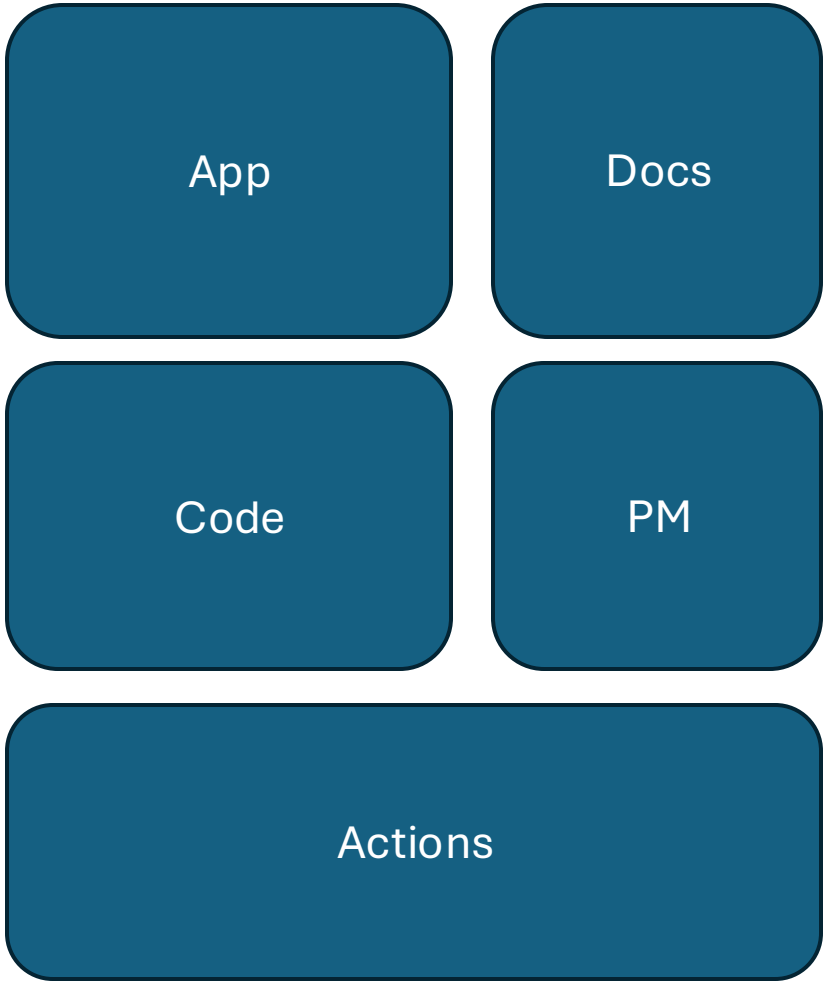




GitHub as Opensource Infrastructure

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

[Preview] README.md

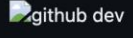
What is this?

> 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 main 0 0

Layout: U.S.

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