# Lecture08 - "Fun" with indices

Kevin Bonham, PhD

2021-07-02

## Outline

# Final project: Code + analysis

## Final Contents

- Code repository with resuable functions (largely built from labs and assignments)
    - including documentation
    - including test suite
- Analysis repository with details, descriptions of code, and plots
- Default project: sequence analysis of Sars-CoV genomes

## Earlier components

- If proposing alternate project: Analysis proposal
- Analysis plan
- First draft
    - Code, tests, docs should be complete
    - Analysis should be complete

## Dates

### School Deadlines

- Summer Term Final Projects Due: July 23
- Grades Due: July 30

### Project Deadlines

- Proposals for alternate projects Due: July 9
- Analysis plans Due: July 12
- First drafts code / notebooks Due: July 16

Missing these deadlines will have grading consequences

## Confusion between "location" and "thing"

```
julia> myvec = [1.2, 2.3, 3.4]
3-element Vector{Float64}:
 1.2
 2.3
 3.4

julia> x = myvec[2]; # 2 is the index

julia> x # this is the value stored at index 2
2.3
```

## Indices must be integers, values can be anything

```
julia> myvec[1]
1.2

julia> myvec[1.2]
ERROR: ArgumentError: invalid index: 1.2 of type Float64
#...
julia> othervec = ["something", 'A', 2.2];

julia> map(typeof, othervec)
3-element Vector{DataType}:
 String
 Char
 Float64
```

## Specialized functions can find things inside vectors

```
julia> newvec = rand(5)
5-element Vector{Float64}:
 0.516023786589465
 0.4465775523061499
 0.2178789287837185
 0.08900106348786951
 0.7016481961587768

julia> findfirst(<(0.5), newvec)
2

julia> findall(<(0.5), newvec)
3-element Vector{Int64}:
 2
 3
 4

julia> newvec[findall(<(0.5), newvec)] # index based on result
3-element Vector{Float64}:
 0.4465775523061499
```

## Mixing code, results, and descriptions

- can use comments, but they have limited expressiveness
- using “notebook” environments allows including results “inline”
- Many options for notebooks,
  - markdown (R Markdown, Weave.jl)
  - Jupyter notebooks
  - Pluto.jl

## For scientific coding, code is usually ad-hoc

- Can't write unit tests for a specific plot
- But you *can* for the plot function itself
- Functions, packages, etc, are like "protocols"
- Use code notebooks for "experiments"

# Alignment tracing for NW and SW

- start from $M_{(i,j)}$ where $i$ and $j$ are
  - the last indices in 1st and 2nd dimension for NW
  - the indices for the matrix with the maximum score in SW
- Check the score from
  1. $M_{(i,j-1)}$ (cell to the left), a gap score
  2. $M_{(i-1,j)}$ (cell above), a gap score
  3. $M_{(i-1,j-1)}$ (cell from diagonal), a match or mismatch
- If any match your current cell, push correct characters to alignments
  1. push gap to seq1, character at $j$ to seq2
  2. push character at $i$ to seq1, gap to seq2
  3. push character at $i$ to seq1, character at $j$ to seq2
- update indices

## Special considerations

- Be mindful of what happens when you hit the first row or first column
  - $i - 1$ or $j - 1$ may throw bounds error
- When should your loop stop?
  - It will be different for Needleman-Wunsch than for Smith-Waterman