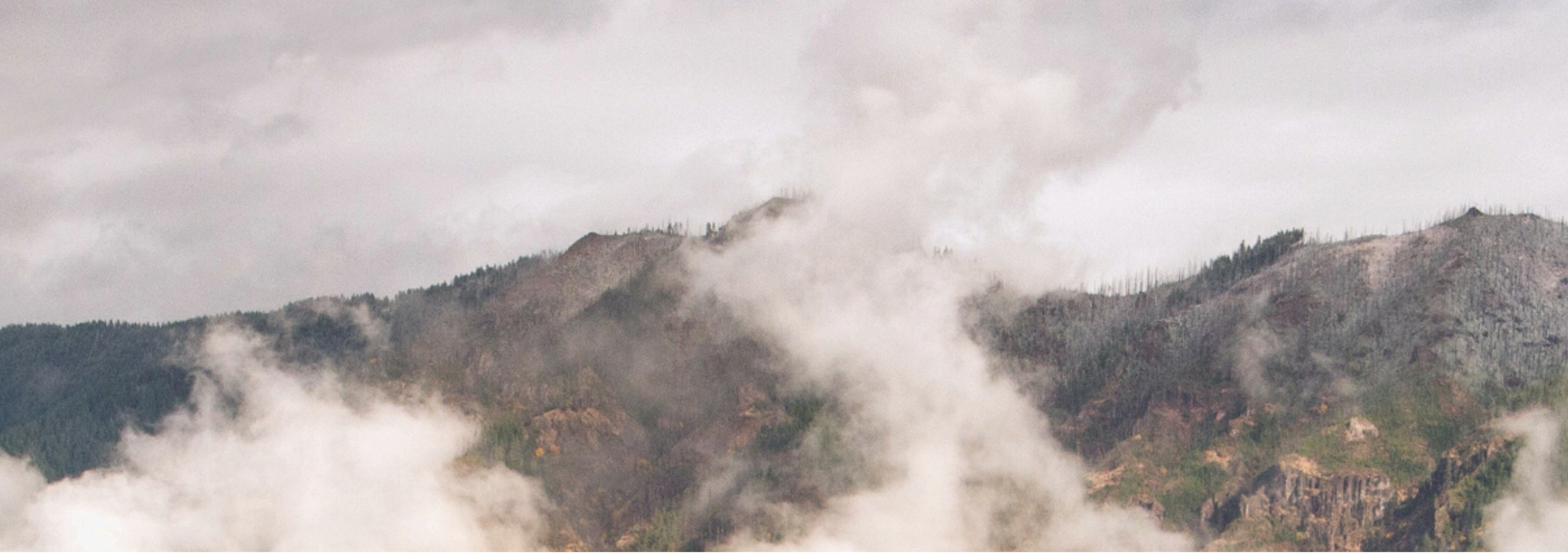
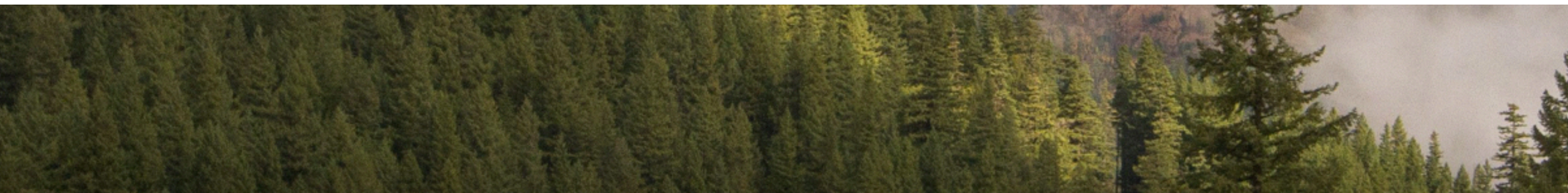


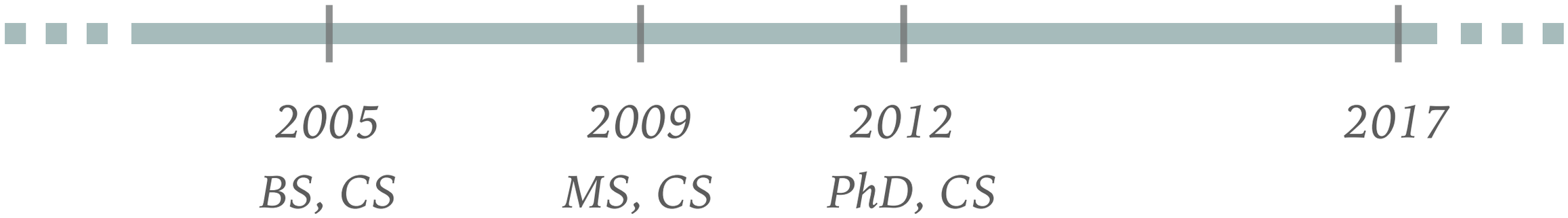
TEACHING AND USING R

A CS Perspective

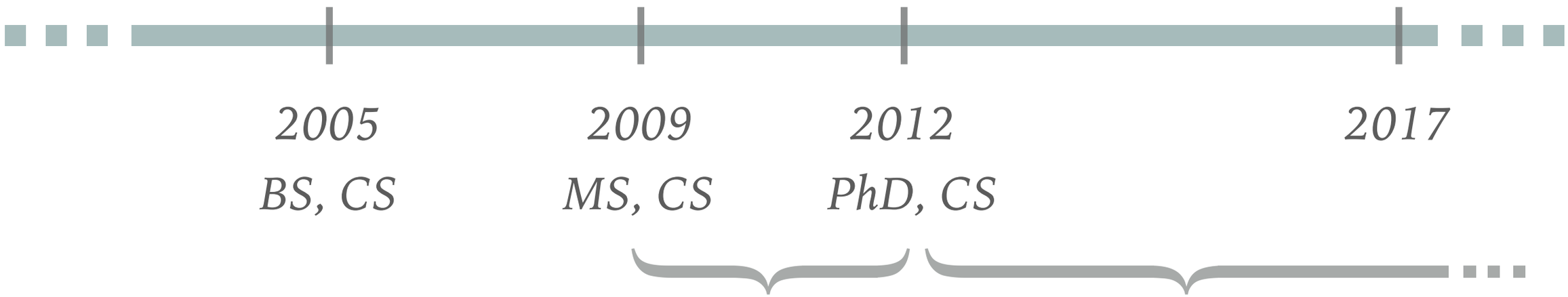
Shawn T. O'Neil, OSU



ABOUT ME



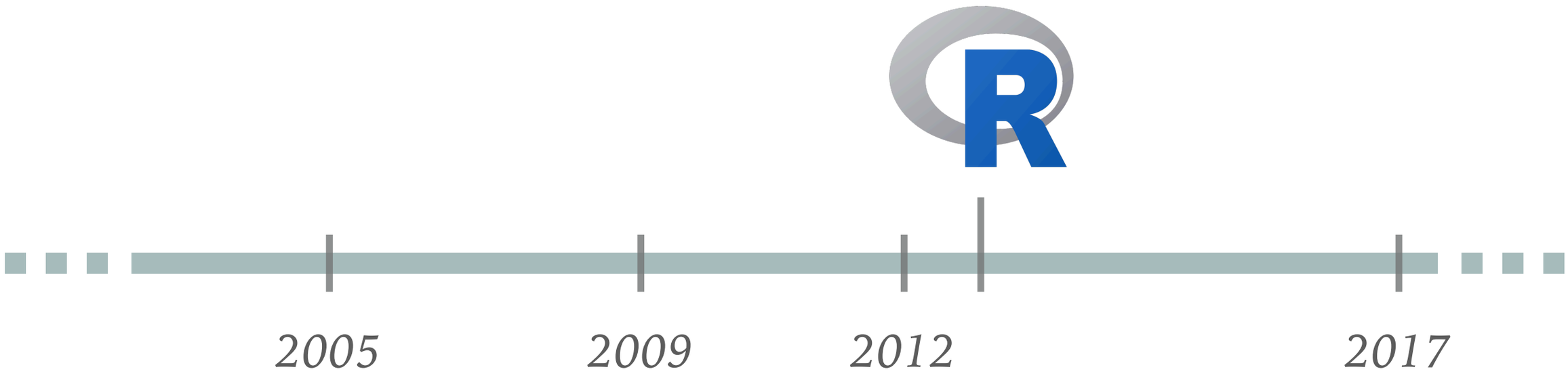
ABOUT ME



CENTER FOR
GENOME RESEARCH &
BIOCOMPUTING

Oregon State
University

ABOUT ME



CENTER FOR
GENOME RESEARCH &
BIOCOMPUTING

Oregon State
University

I'm a newcomer to R!

LEARNING R

- Vectorization
- 1-based indexing

LEARNING R

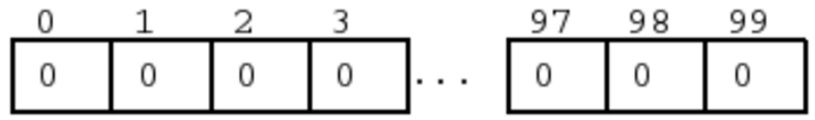
- Vectorization
- 1-based indexing

- Vectors
- Data Frames
- Matrices
- Lists
- Named Types
- .....

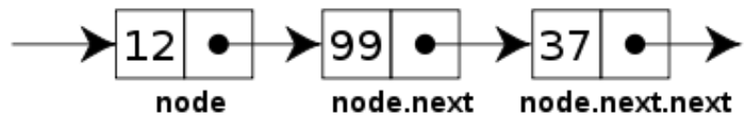
LEARNING R

- Vectorization
- 1-based indexing
- Vectors
- Data Frames
- Matrices
- Lists
- Named Types
- ……

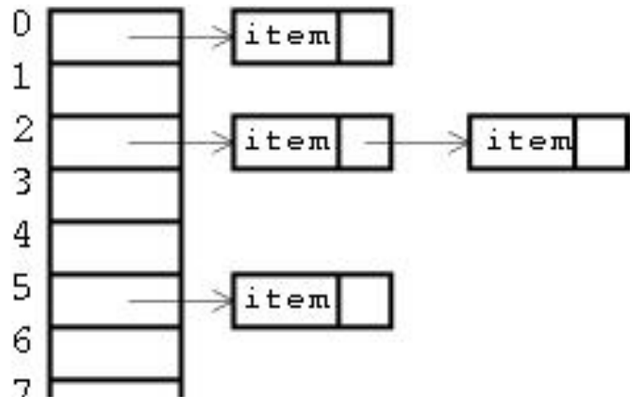
- Arrays?



- Linked Lists?



- Hash Tables?

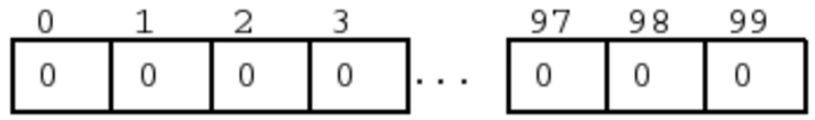


- ……

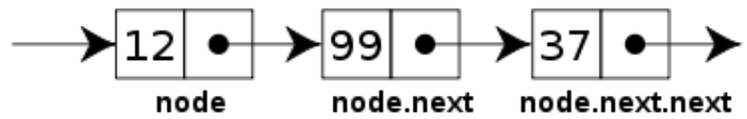
LEARNING R

- Vectorization
- 1-based indexing
- Vectors
- Data Frames
- Matrices
- Lists
- Named Types
- ……

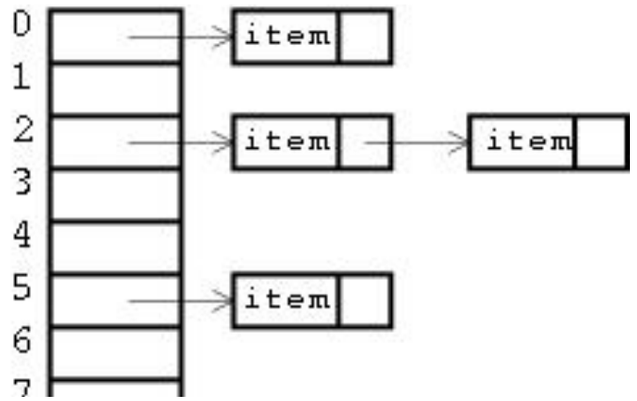
- Arrays?



- Linked Lists?



- Hash Tables?



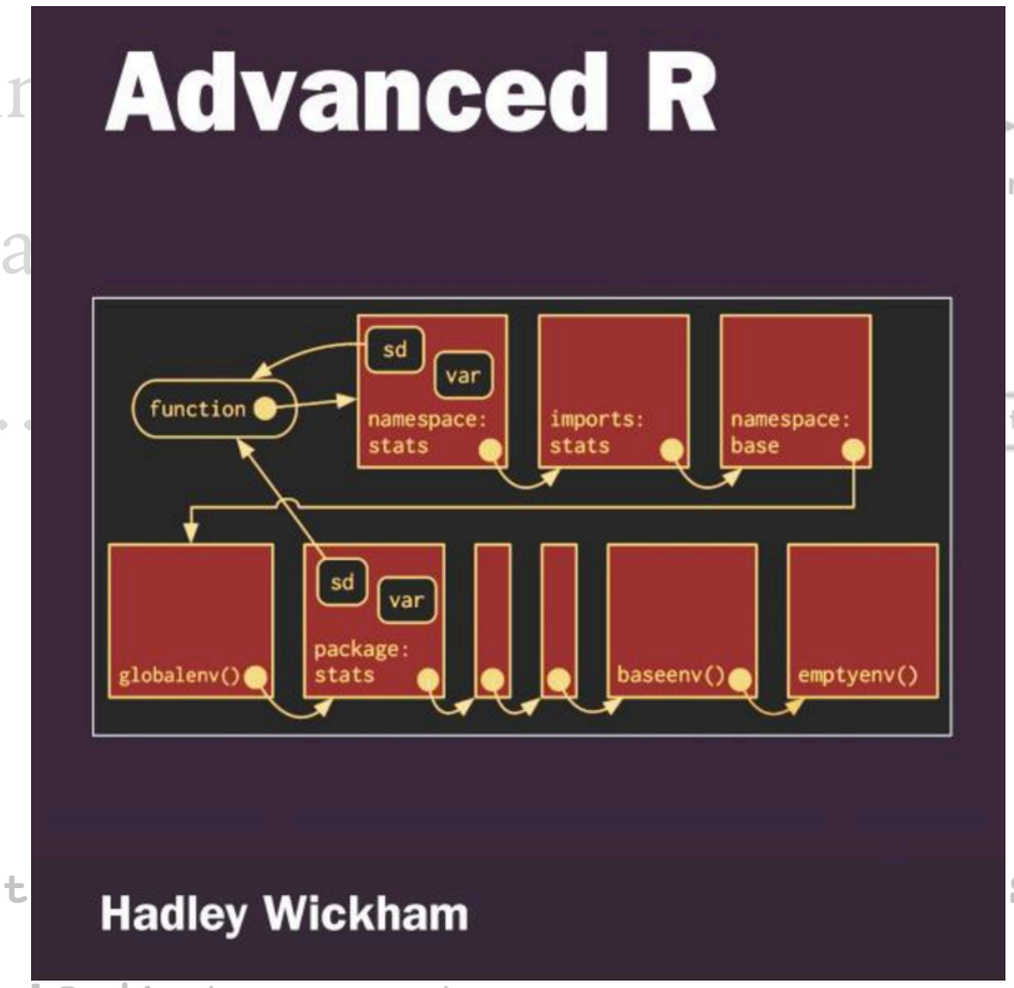
- ……

```
SEXP attribute_hidden do_lengths(SEXP call, SEXP op, SEXP
{
    checkArity(op, args);
    SEXP x = CAR(args), ans;
    R_xlen_t x_len, i;
    int *ans_elt;
    int useNames = asLogical(CADR(args));
    if (useNames == NA_LOGICAL)
   error( ("invalid '%s' value"), "use.names");
```

LEARNING R

- Vectorization
- 1-based indexing
- Vectors
- Data Frames
- Matrices
- Lists
- Named Types
- ……

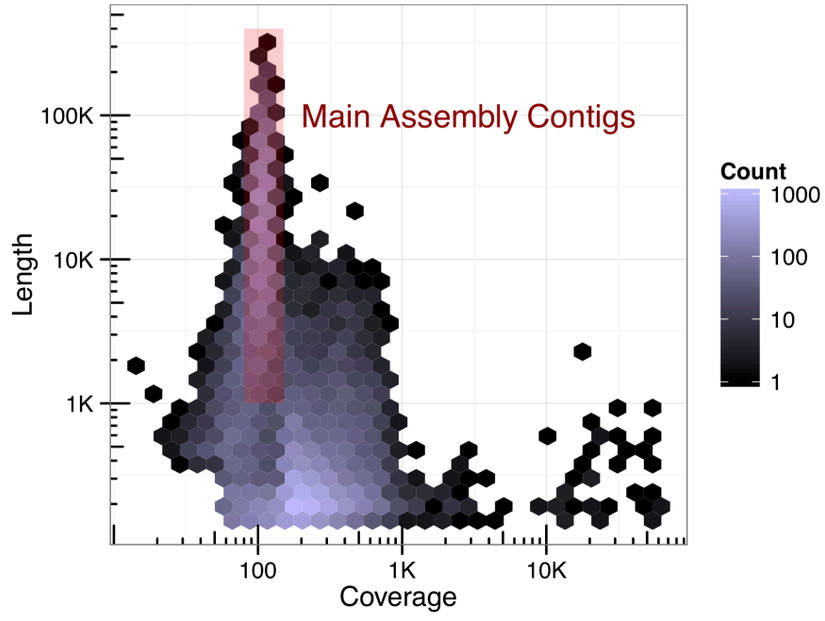
- Arrays?
- Lin
- Ha
- ….




```
SEXP att                                                    SEXP op, SEXP
{
    checkArity(op, args);
    SEXP x = CAR(args), ans;
    R_xlen_t x_len, i;
    int *ans_elt;
    int useNames = asLogical(CADR(args));
    if (useNames == NA_LOGICAL)
   error( ("invalid '%s' value"), "use.names");
```

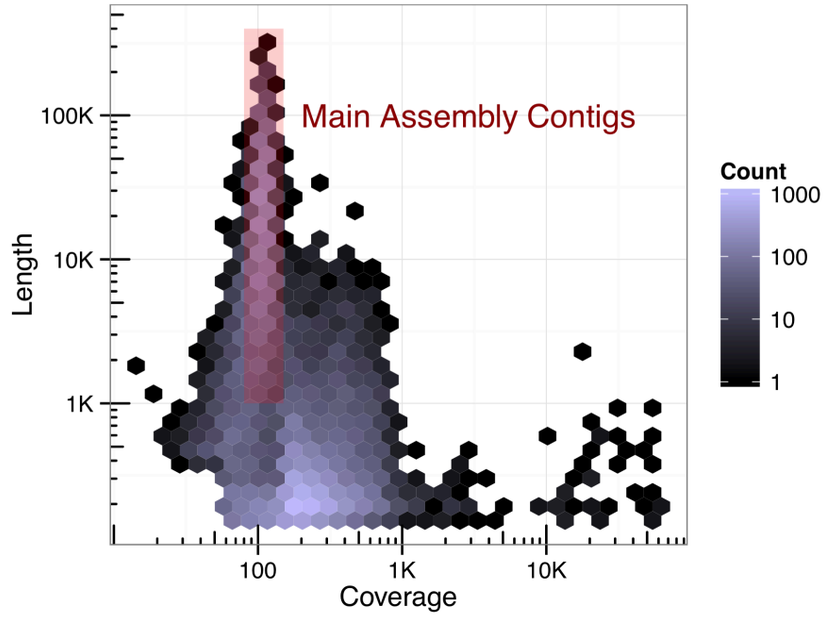
TEACHING R

- Data Programming in R
  - Basics, Practicals
  - (Almost) No Stats
  - S3, Split/Apply/Combine, some functional vs procedural

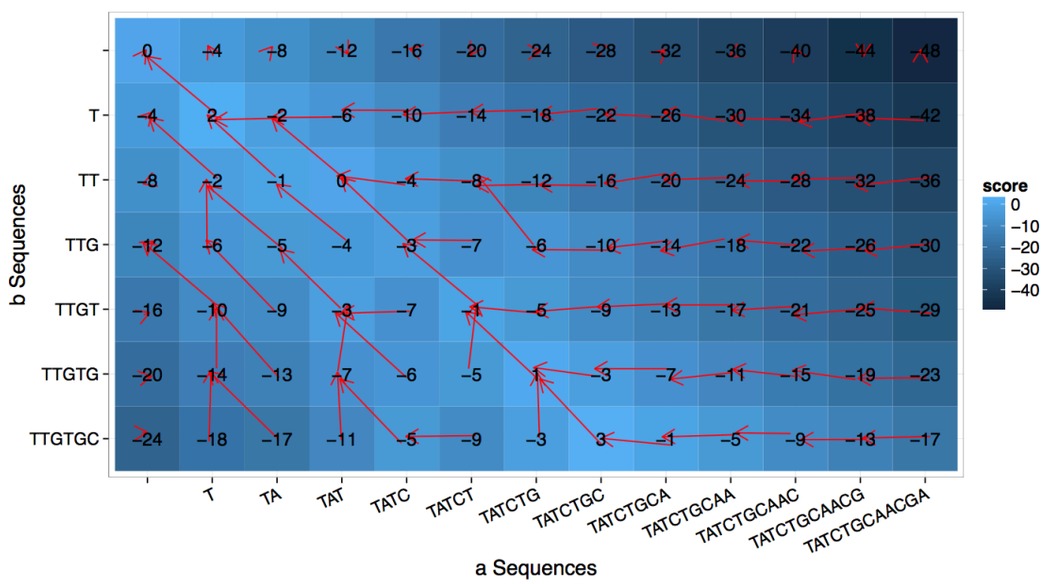


TEACHING R

- Data Programming in R
  - Basics, Practicals
  - (Almost) No Stats
  - S3, Split/Apply/Combine, some functional vs procedural



- Recursion and Dynamic Programming for Sequence Analysis
  - Data Structures
  - CS Techniques
  - Bioinformatics



DATA PROGRAMMING IN R

Vectorized
(element-by-element)

Pass-by-value
(functions get copies of data)

Functional
(pass functions around as data - a lot)

DATA PROGRAMMING IN R

Vectorized
(element-by-element)

Pass-by-value
(functions get copies of data)

Functional
(pass functions around as data - a lot)

not Vectorized

not Pass-by-value

not (very) Functional
(sorry)

DATA PROGRAMMING IN R

Vectorized
(element-by-element)

Pass-by-value
(functions get copies of data)

Functional
(pass functions around as data - a lot)

not Vectorized

not Pass-by-value

not (very) Functional
(sorry)

1. Ints, Floats
2. Strings, Booleans
3. Lists, Loops
4. File I/O
5. If-statements
6. Functions
7. Objects
8. APIs, Modules etc.

DATA PROGRAMMING IN R

Vectorized
(element-by-element)

Pass-by-value
(functions get copies of data)

Functional
(pass functions around as data - a lot)

not Vectorized

not Pass-by-value

not (very) Functional
(sorry)

1. Vectors
    1. Character, Logical, Numeric (Double), Integer
2. Vectorized Operations
    1. Selection & Replacement
    2. Vectorization
3. Functions
4. Lists
    1. Attributes
    2. Named types
5. Data Frames, Matrices
6. Factors

1. Ints, Floats
2. Strings, Booleans
3. Lists, Loops
4. File I/O
5. If-statements
6. Functions
7. Objects
8. APIs, Modules etc.

DATA PROGRAMMING IN R

Vectorized
(element-by-element)

Pass-by-value
(functions get copies of data)

Functional
(pass functions around as data - a lot)

not Vectorized

not Pass-by-value

not (very) Functional
(sorry)

1. Vectors
    1. Character, Logical, Numeric (Double), Integer
2. Vectorized Operations
    1. Selection & Replacement
    2. Vectorization
3. Functions
4. Lists
    1. Attributes
    2. Named types
5. Data Frames, Matrices
6. Factors
7. Split/Apply/Combine
8. Data munging & tidiness
9. Procedural techniques
10. S3 Objects, Generics & Dispatch
11. Some plotting & ggplot2

1. Ints, Floats
2. Strings, Booleans
3. Lists, Loops
4. File I/O
5. If-statements
6. Functions
7. Objects
8. APIs, Modules etc.

RECURSION & DYNAMIC PROGRAMMING (CS STUFF) IN R

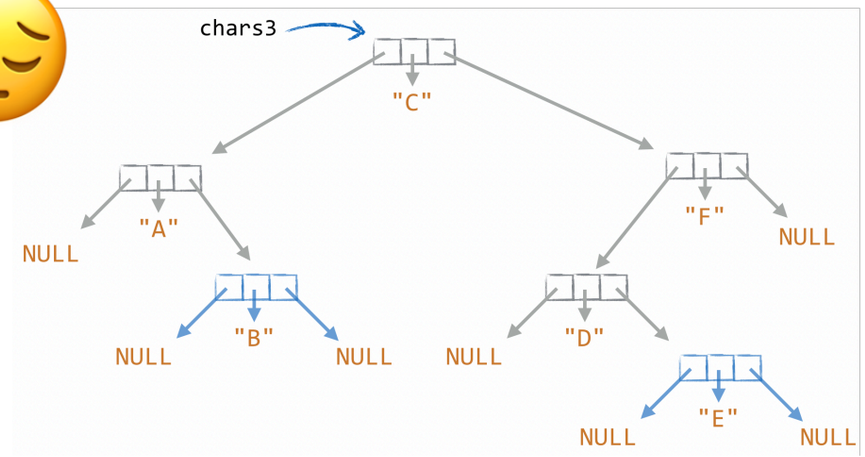
- History:
  - Biologists utilize deeply beautiful algorithms

RECURSION & DYNAMIC PROGRAMMING (CS STUFF) IN R

- History:
  - Biologists utilize deeply beautiful algorithms
  - But generally aren't exposed to that beauty 😔
  - They do tend to know and like R…

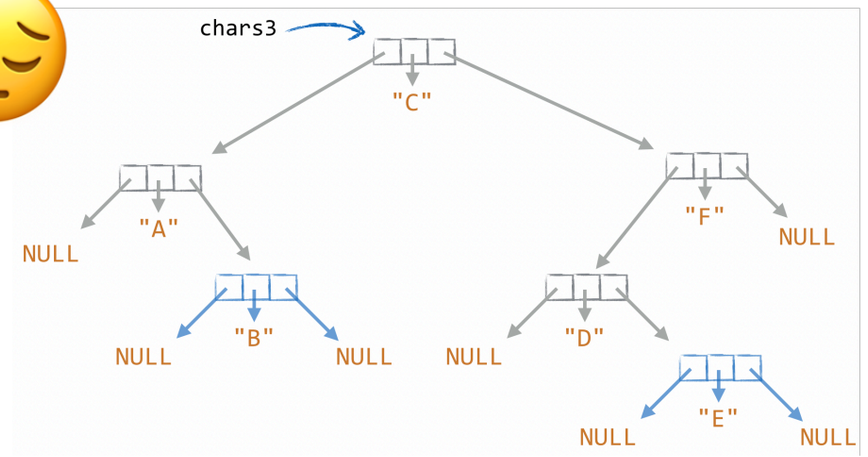
RECURSION & DYNAMIC PROGRAMMING (CS STUFF) IN R

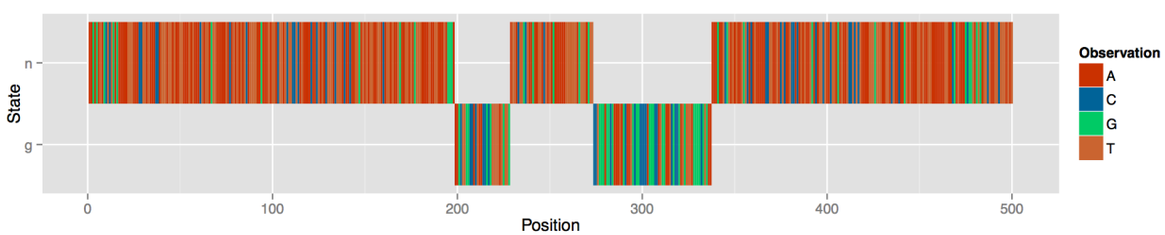
- History:
  - Biologists utilize deeply beautiful algorithms
  - But generally aren't exposed to that beauty
  - They do tend to know and like R…
- Benefits of R for this kind of material:
  - Pure functions, functional concepts (recursion! trees!)

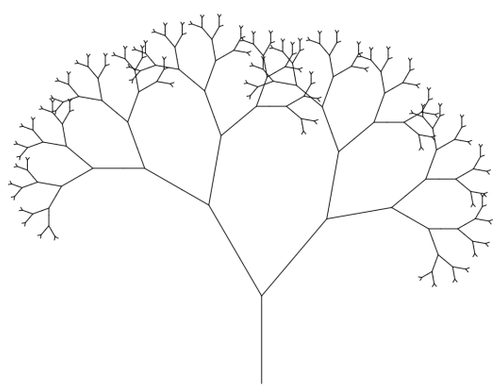


RECURSION & DYNAMIC PROGRAMMING (CS STUFF) IN R

- History:
  - Biologists utilize deeply beautiful algorithms
  - But generally aren't exposed to that beauty
  - They do tend to know and like R…
- Benefits of R for this kind of material:
  - Pure functions, functional concepts (recursion! trees!)
  - Visualization (`ggplot2`, `TurtleGraphics`)

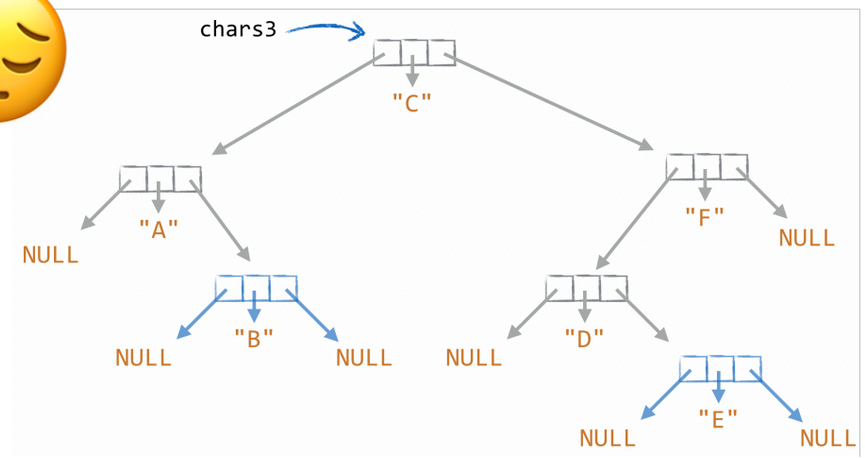


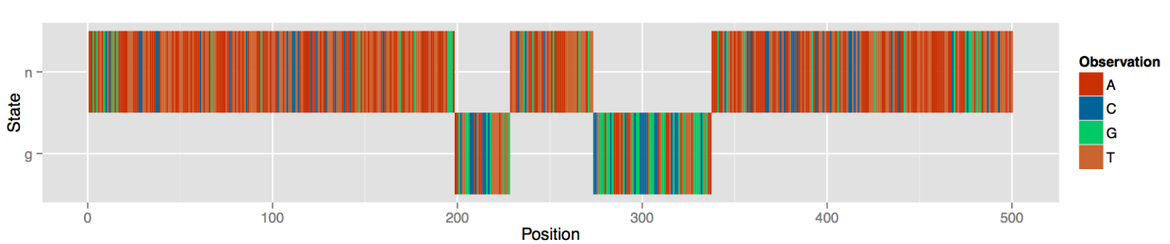


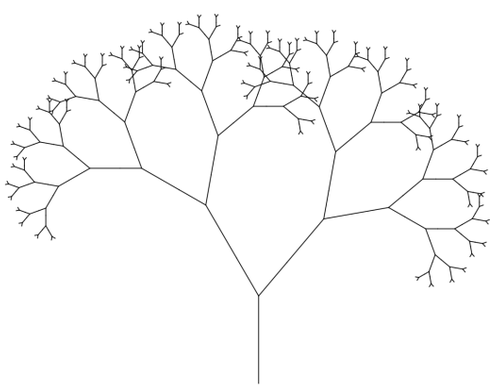


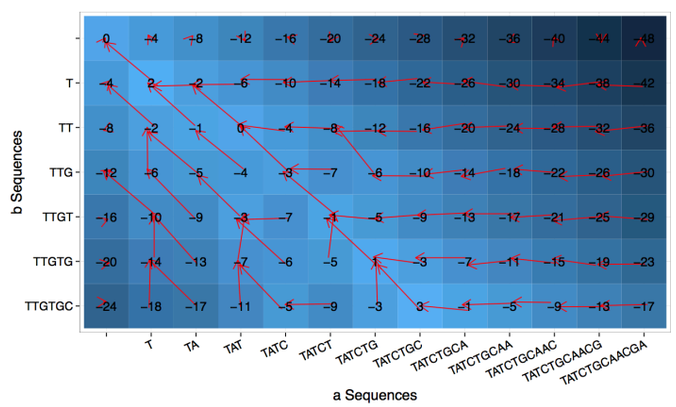
RECURSION & DYNAMIC PROGRAMMING (CS STUFF) IN R

- History:
  - Biologists utilize deeply beautiful algorithms
  - But generally aren't exposed to that beauty
  - They do tend to know and like R…
- Benefits of R for this kind of material:
  - Pure functions, functional concepts (recursion! trees!)
  - Visualization (`ggplot2`, `TurtleGraphics`)
  - Not *too* pure… (matrices and loops for dynamic programming, globals for memoization)



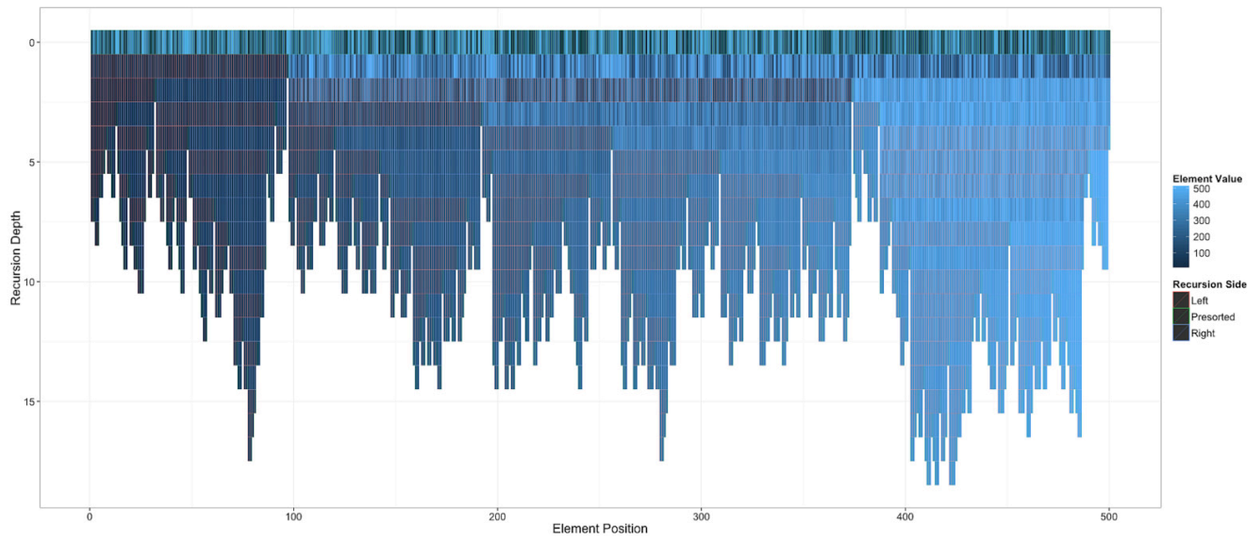






TOPICS

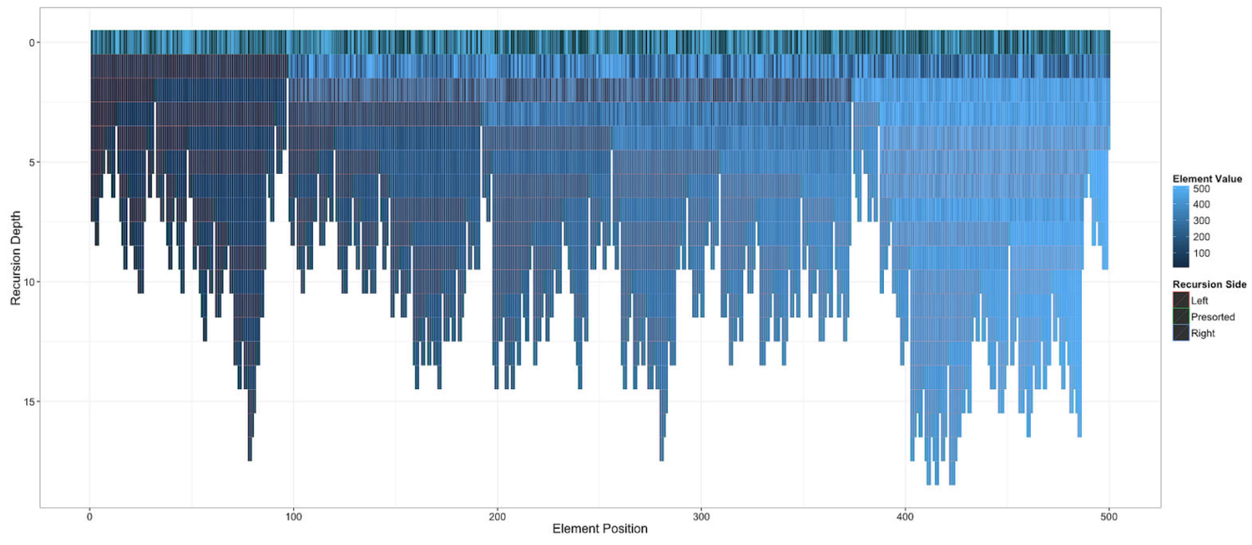
- Review R syntax, esp. lists & functions
- Linked lists, binary search trees
- Sorting & searching
- Stacks and Queues
  (rabbit hole: *pure functional data structures* ⇒ `rstackdeque`)



Visualizing Quicksort

TOPICS

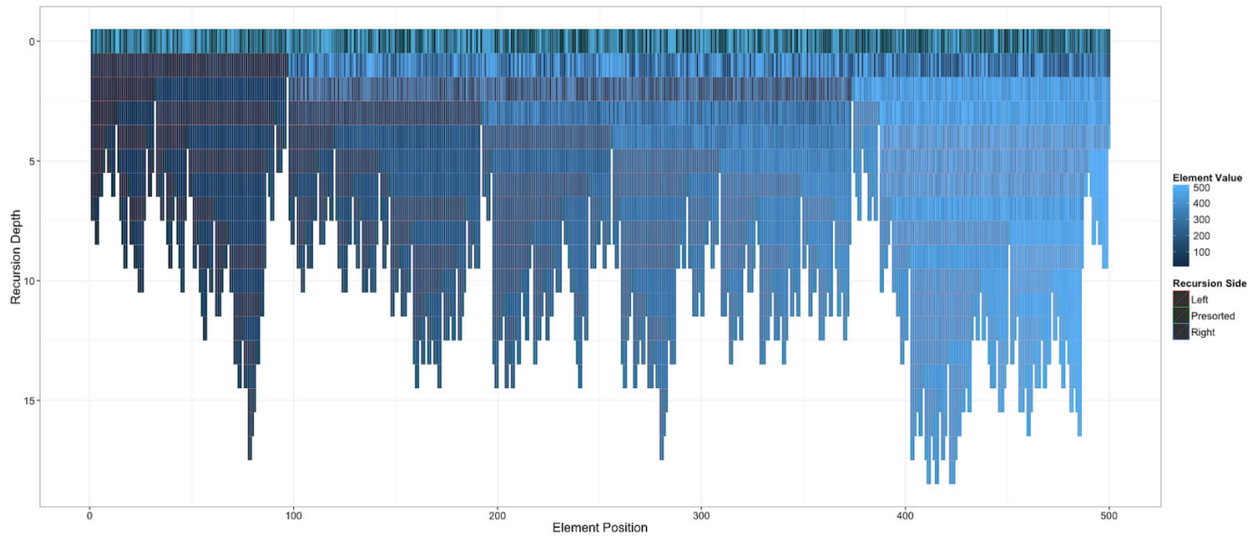
- Review R syntax, esp. lists & functions
- Linked lists, binary search trees
- Sorting & searching
- Stacks and Queues
  (rabbit hole: *pure functional data structures* ⇒ `rstackdeque`)
- Call stack
- Memoization (caching), Hashes
- Dynamic Programming



Visualizing Quicksort

TOPICS

- Review R syntax, esp. lists & functions
- Linked lists, binary search trees
- Sorting & searching
- Stacks and Queues
  (rabbit hole: *pure functional data structures* ⇒ `rstackdeque`)
- Call stack
- Memoization (caching), Hashes
- Dynamic Programming
- Fast matching (suffix arrays)
- Global sequence alignment
  (recursive ⇒ memoized ⇒ dyn. prog.)
- Local alignment
- Hidden Markov Models, Viterbi
- Recursive Drawing, L-Systems



Visualizing Quicksort

CHALLENGES & OPPORTUNITIES

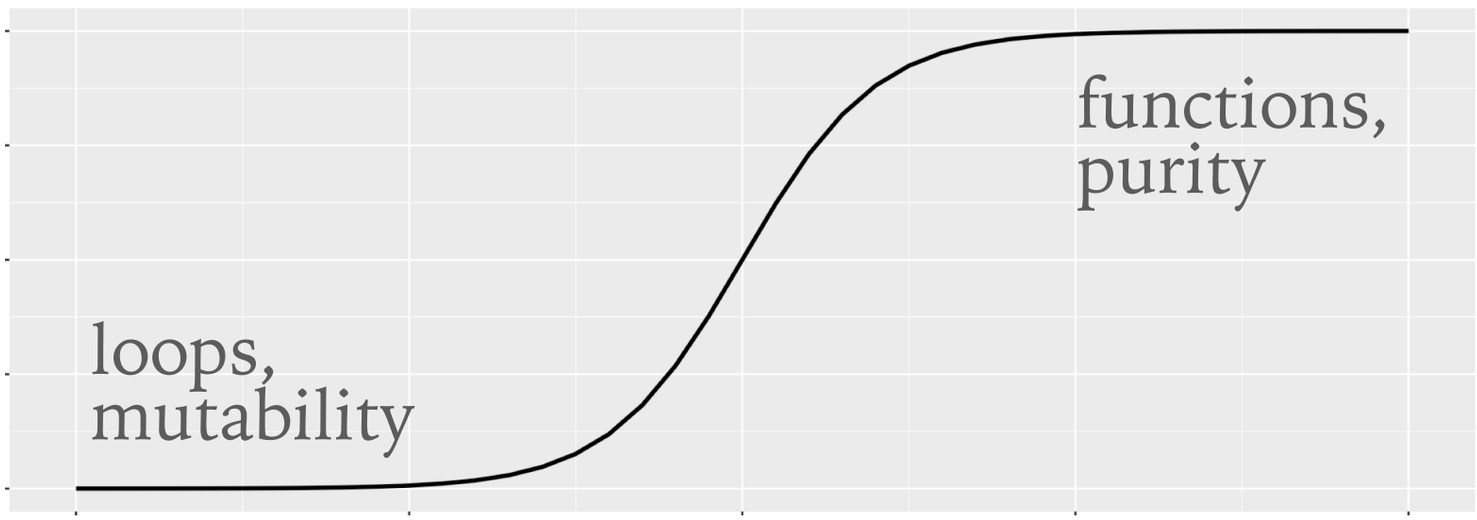
- Still a lot of syntax
  `$, [[]], [], =, <-, DSLs, …`
- Dynamic Typing
- `"ACTAG"` vs
  `c("A", "C", "T", "A", "G")`
- Awkward conversions
  (stack ⇒ list ⇒ vector ⇒ df col)

CHALLENGES & OPPORTUNITIES

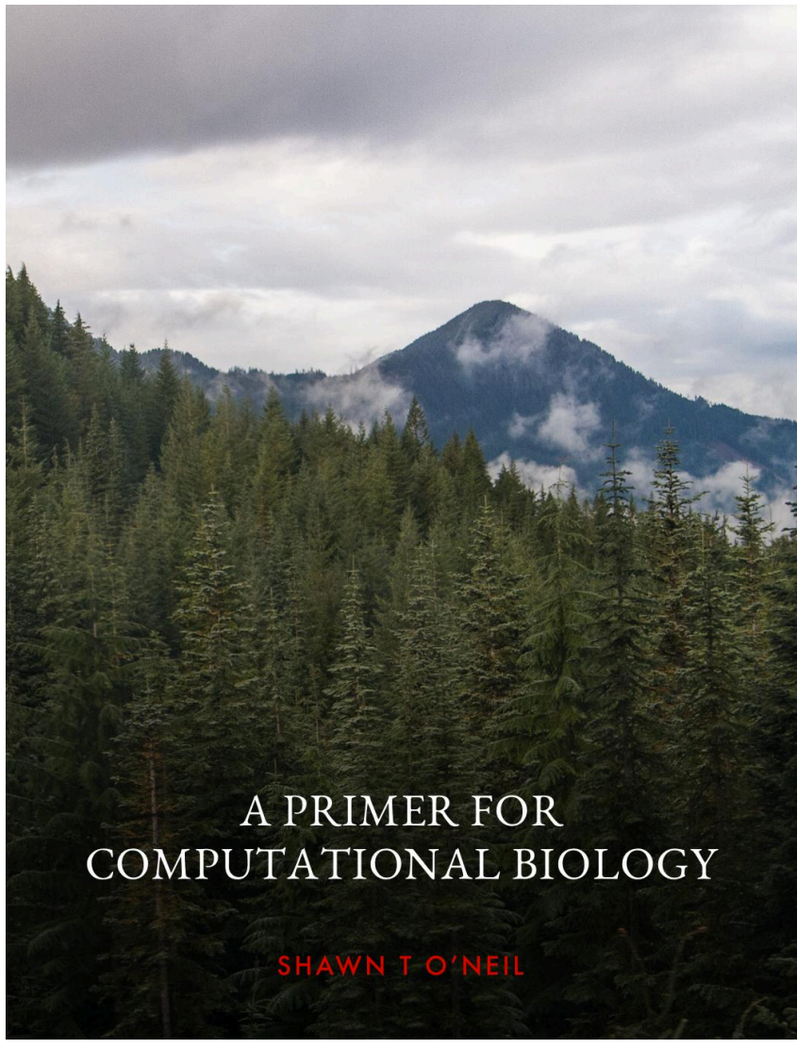
- Still a lot of syntax
  `$`, `[[]]`, `[]`, `=`, `<-`, `DSLs`, …
- Dynamic Typing
- `"ACTAG"` vs
  `c("A", "C", "T", "A", "G")`
- Awkward conversions
  (stack ⇒ list ⇒ vector ⇒ df col)
- Fun with Data &
  Process Visualization
  (want to try `gganimate`, `ggtree` et al.)

CHALLENGES & OPPORTUNITIES

- Still a lot of syntax
  `$`, `[[]]`, `[]`, `=`, `<-`, `DSLs`, …
- Dynamic Typing
- `"ACTAG"` vs
  `c("A", "C", "T", "A", "G")`
- Awkward conversions
  (stack ⇒ list ⇒ vector ⇒ df col)
- Fun with Data &
  Process Visualization
  (want to try `gganimate`, `ggtree` et al.)
- Explore the inflection point between procedural and functional programming
- And data and process



RESOURCES



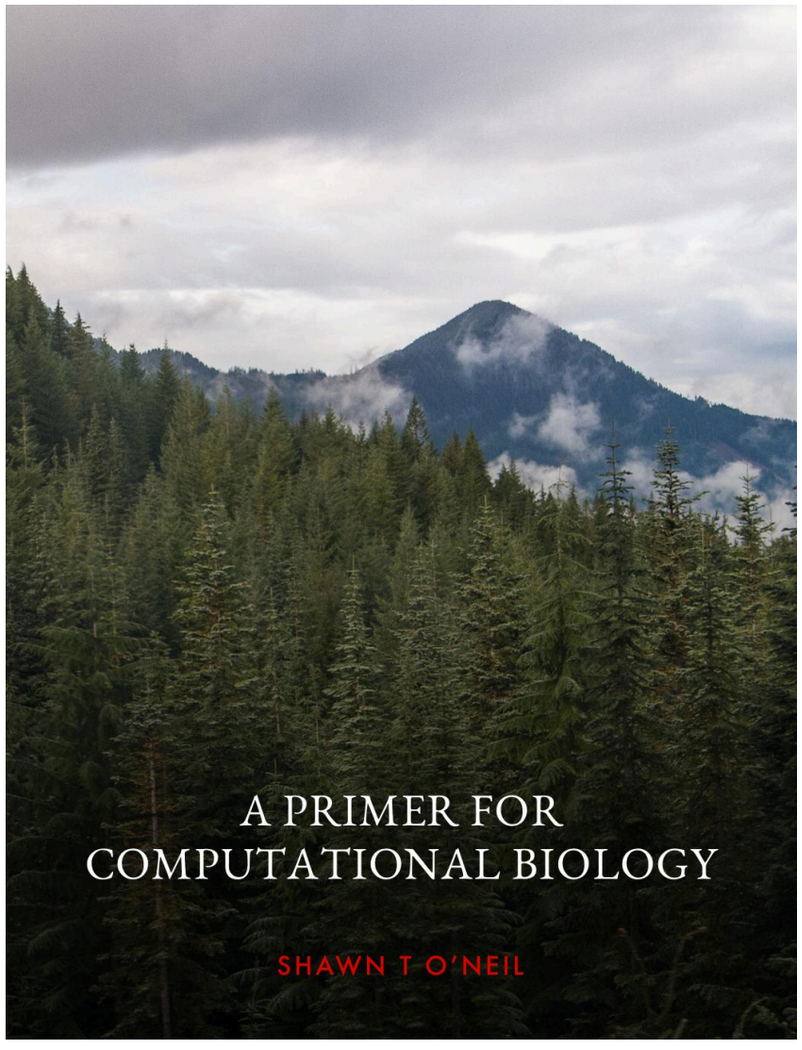
Basic Linux, Python, R

Open Access

teaching.cgrb.oregonstate.edu/CGRB/oneil/primer/

bit.ly/2qGZDAw

RESOURCES

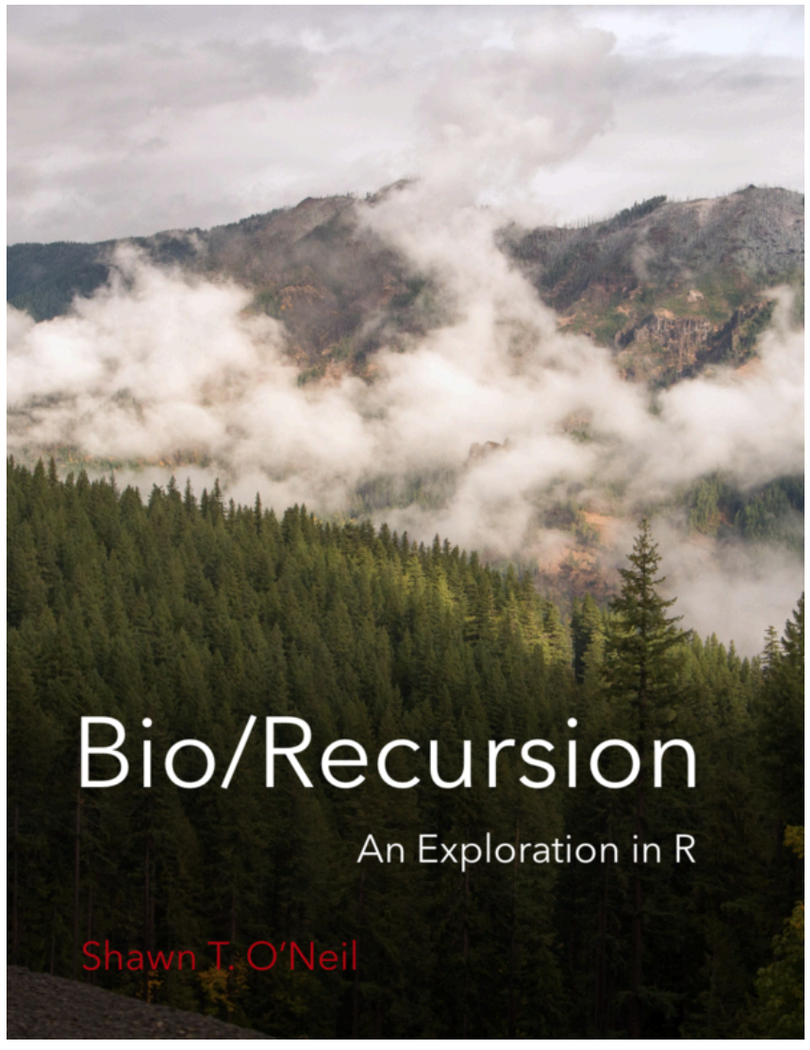


Basic Linux, Python, R

Open Access

`teaching.cgrb.oregonstate.edu/CGRB/oneil/primer/`

`bit.ly/2qGZDAw`



Recursion etc.

Leanpub, Coupon:

`leanpub.com/biorecursion/c/cascadiarconf`

`bit.ly/2qGznq8`