Math For Rails Programmers

@evanburchard

Overview

- Intro
- Where is the math in Ruby?
- What is Ruby missing?
- Interview Math
- Syntax from equations and academic papers

Overview: !Doing

- Defining Math
- History of Math vs. Programming vs. CS
- Deep math calculations. Likely some unfamiliar material, but nothing we need a graphing calculator for.

Overview: Procedure

- I'll have "Q" when I have questions for you.

Overview: Procedure

- I'll have "Q" when I have questions for you.
- Stop me anytime if you have questions.

Overview: Procedure

- I'll have "Q" when I have questions for you.
- Stop me anytime if you have questions.
- If you miss anything or want resources, the slides will be at github.com/evanburchard/math_for_rails
- You can tweet at me @evanburchard

Hi, I'm Evan

- I'm a Rails programmer.

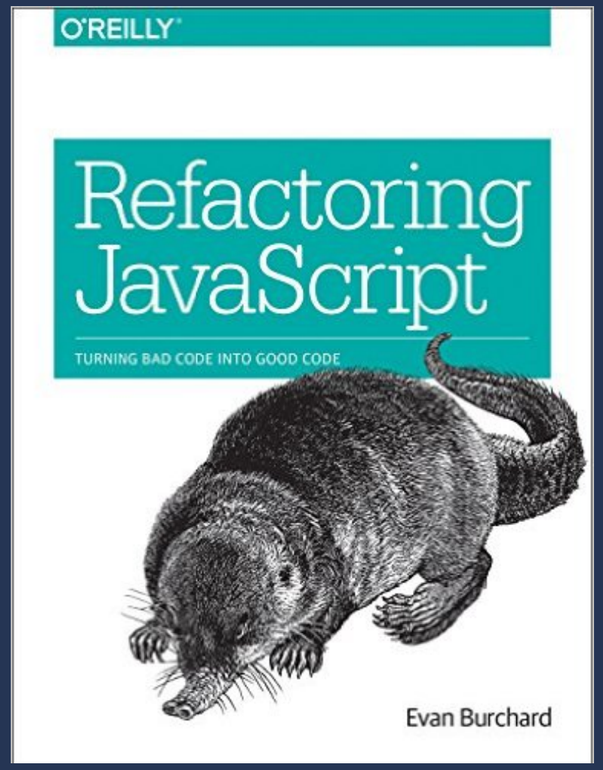
Hi, I'm Evan

- I'm a Rails programmer.
- I write about programming:

HTML5 Game Developer's Cookbook

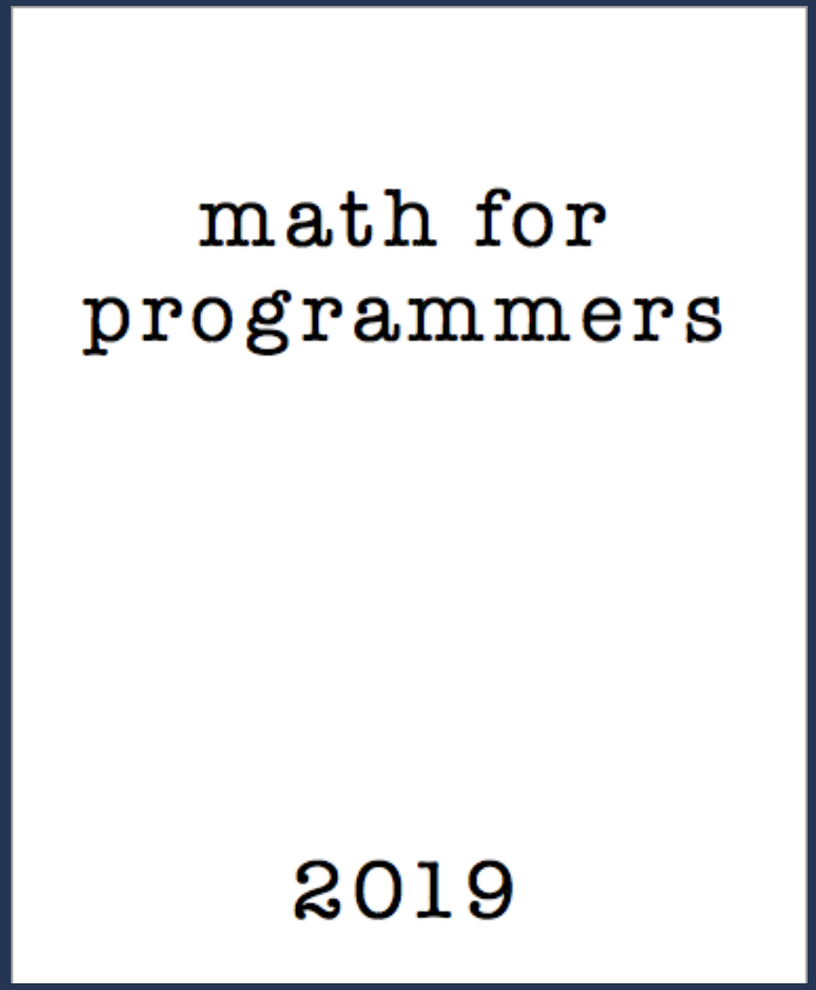
@evanburchard

Refactoring JavaScript



@evanburchard

Next up...



Hi, I'm Evan

- I'm a Rails programmer.
- I write about programming.
- On the side, I've been working on a site:

Hi, I'm Evan

- I'm a Rails programmer.
- I write about programming.
- On the side, I've been working on a site:

jobpika.com

@evanburchard

But anyways

This workshop is about math.

Math

Math

- I didn't really love math.

Math

- I didn't really love math.
- Just kept getting frustrated by not understanding.

Math

- I didn't really love math.
- Just kept getting frustrated by not understanding.
- Kept finding contexts where more math would help.

Q: What's your math experience?

Q: What are you interested in?


One last job
@evanburchard

One last job

@evanburchard

One last job

After the interview, we'll never need mathy things again, right?

One last job

After the interview, we'll never need mathy things again, right?

If that's true (and it is definitely just unnecessary and uncreative gatekeeping for some jobs), as programmers, why should we care about math?

One last job

After the interview, we'll never need mathy things again, right?

If that's true (and it is definitely just unnecessary and uncreative gatekeeping for some jobs), as programmers, why should we care about math?

(We'll talk more about interviews later)

Math is actually...

@evanburchard

The long game

1. Interesting stuff

The long game

1. Interesting stuff
2. Other types of programming (Visualization, Games, Functional Programming, Machine Learning/Stats)

The long game

1. Interesting stuff
2. Other types of programming (Visualization, Games, Functional Programming, Machine Learning/Stats)
3. Actually improve Rails programming

Math In Ruby (Explicit)

Where can we find math in Ruby?

Where math lives (Math module)

- Trig Functions (including hyperbolic functions)
- Cube root, Square root
- Error Function, complimentary Error Function
- exp (e**x)
- frexp exponent and fraction with base 2

Q: Anyone used these? How about the hyperbolic functions?

Where math lives (Math module)

- Geometry (hypot)
- ldexp (long double exponent)
- log, log10, log2
- gamma function (kind of like factorial) and logarithmic gamma (lgamma)

Q: Anyone used these?

Where Math lives in Ruby (2)

- Math.constants
- FixNum, Float and BigNum, Numeric
- "unary" (-, !, etc.)
- "binary" operators (+, *, etc.)
- Various utilities for arithmetic (eg. Time and Date)

Q: Why is the - in -1 called the "unary" minus operator?

Interesting thing!

Arity vs. fixity vs. precedence

Interesting thing!

Arity vs. fixity vs. precedence

Q: What is the difference?

Interesting thing!

Arity vs. fixity vs. precedence

- Arity: The number of arguments (nullary, unary, binary, ternary, veriadic, n-ary, etc.)

Interesting thing!

Arity vs. fixity vs. precedence

- Arity: The number of arguments (nullary, unary, binary, ternary, veriadic, n-ary, etc.)
- Fixity: The position of the operator (prefix, infix, postfix)

Interesting thing!

Arity vs. fixity vs. precedence

- Arity: The number of arguments (nullary, unary, binary, ternary, veriadic, n-ary, etc.)
- Fixity: The position of the operator (prefix, infix, postfix)
- Precedence: The sequence of execution (eg. PEMDAS)

Interesting thing!

Arity vs. fixity vs. precedence

In Ruby, I mostly define functions/methods. In some languages, operator/function creation are more exposed and similar.

Defining our own operators

Not to go too deep here, but in Ruby, we can define our own operators, but methods are a little more familiar.

* Sidenote, operators are methods in Ruby.

Q: Which is the unary minus?

```
3.method(:-)
3.method(:-@)
```

How to find where methods come from:

- true.method(:!@).owner
- 3.method(:-).owner
- metho(:puts).owner

How to find what methods an object has:

@evanburchard

The method method

- some.method.owner will tell you where it came from
- 3.method(:+).owner
- send(:method_name)
- plus_three = 3.method(:+)
- plus_three.call(2)
- [1, 2, 4].map(&plus_3)

Where Math lives in Ruby (3)

- Enumerable (also: Arrays, Strings, Hashes, Ranges, Queues)
- Relational "algebra"/relational "calculus" in databases

Where Math lives in Ruby (3)

- Enumerable (also: Arrays, Strings, Hashes, Ranges, Queues)
- Relational "algebra"/relational "calculus" in databases

Discrete Math & Set Theory

Where Math lives in Ruby (4)

- logic operators and binary math

Where Math lives in Ruby (4)

- logic operators and binary math

Discrete Math & Set Theory & Logic

Where Math lives in Ruby (5)

- Functional Programming

Where Math lives in Ruby (5)

- Functional Programming

Discrete Math & Set Theory & Logic & Category Theory

Where Math lives in Ruby (6)

- Matrix

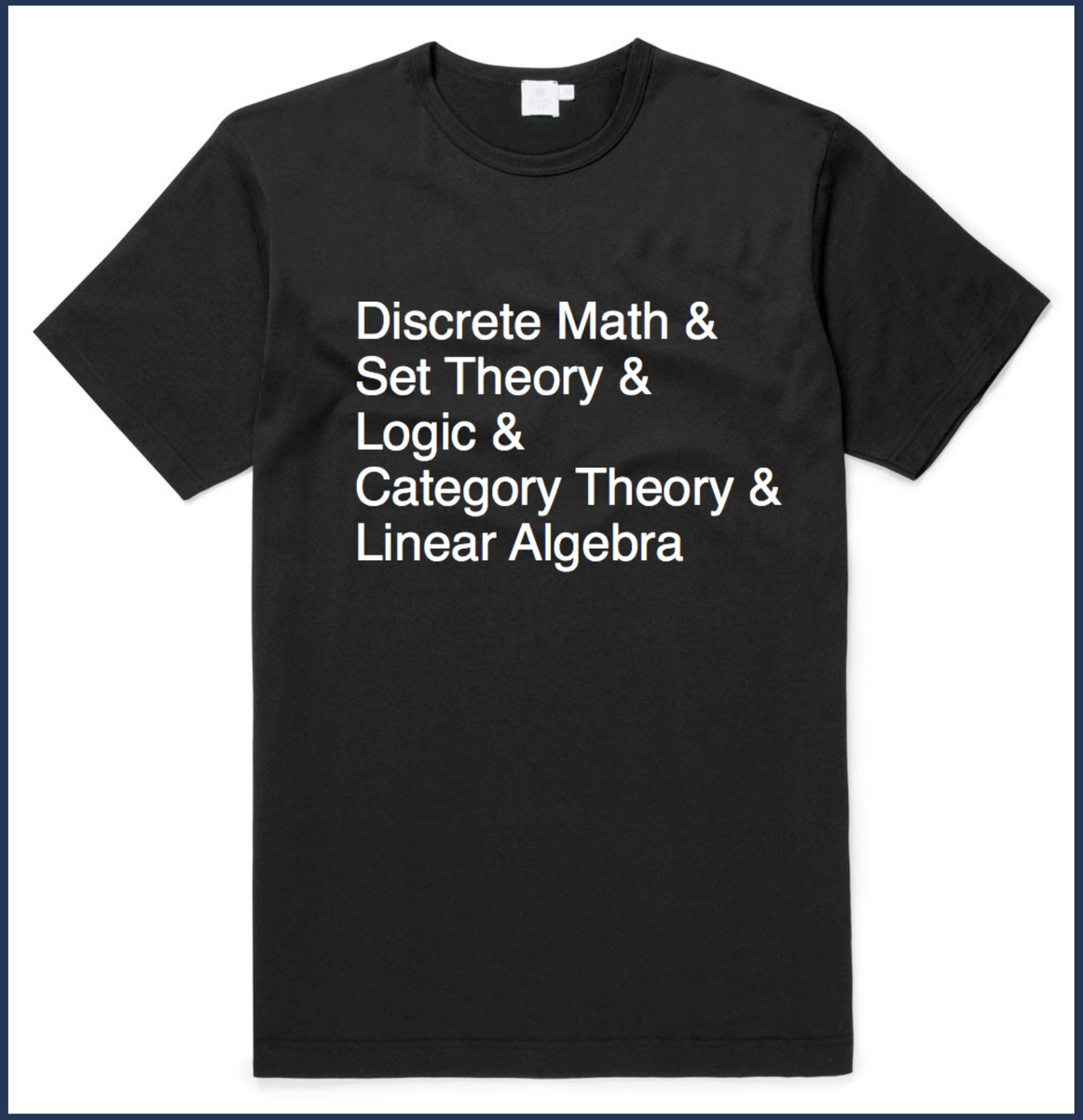
Where Math lives in Ruby (6)

- Matrix...ahem... require 'matrix'

Where Math lives in Ruby (6)

- Matrix...ahem... require 'matrix'

Discrete Math & Set Theory & Logic & Category Theory & Linear Algebra



@evanburchard

Where is calculus?

I don't know.

re. Calculus

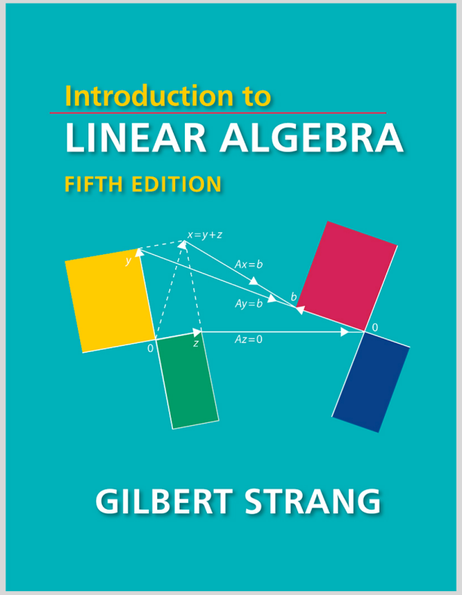
Maybe it's my broken Rubyist brain, but it seems like it's overemphasized in high school as compared with linear algebra or discrete math.

re. Calculus

This might change as CS is more emphasized in High School.
OTOH

WHO WOULD WIN?

instruments too big to mess with



one matrixy boy

Studying math through Ruby

@evanburchard

Studying math through Ruby

Ruby is a pretty good tool for exploring all of these aspects of math (except calculus).

Studying math through Ruby

Ruby is a pretty good tool for exploring all of these aspects of math (except calculus).

But in each case, studying the math directly or through another programming environment might be better.

Studying math through Ruby

R, Python, or actual calculators (or virtual ones) might be better for calculus and linear algebra.

Studying math through Ruby

R, Python, or actual calculators (or virtual ones) might be better for calculus and linear algebra.

C might be better for binary math.

Studying math through Ruby

R, Python, or actual calculators (or virtual ones) might be better for calculus and linear algebra.

C might be better for binary math.

Prolog might be better for logic.

Studying math through Ruby

R, Python, or actual calculators (or virtual ones) might be better for calculus and linear algebra.

C might be better for binary math.

Prolog might be better for logic.

A functional programming language (or even JavaScript) might be better for category theory.

Studying math through Ruby

R, Python, or actual calculators (or virtual ones) might be better for calculus and linear algebra.

C might be better for binary math.

Prolog might be better for logic.

A functional programming language (or even JavaScript) might be better for category theory.

Incidentally, a "calculator" for category theory doesn't exist as

Biggest bang for your buck

Enumerable

Enumerable (& friends)

Anything you can iterate through, sort, and search is a likely target for optimization and refactoring.

Enumerable (& friends)

Anything you can iterate through, sort, and search is a likely target for optimization and refactoring.

Also, you'll find a lot of connections to math through the utilities of enumerable.

Speaking of Optimization...

Interviews

Big O isn't just - "Big Oh No, interviews suck; they suck so bad"

@evanburchard

Elements of Big O

- Time/Space Complexity

Elements of Big O

- Time/Space Complexity
- Analytic Geometry

Elements of Big O

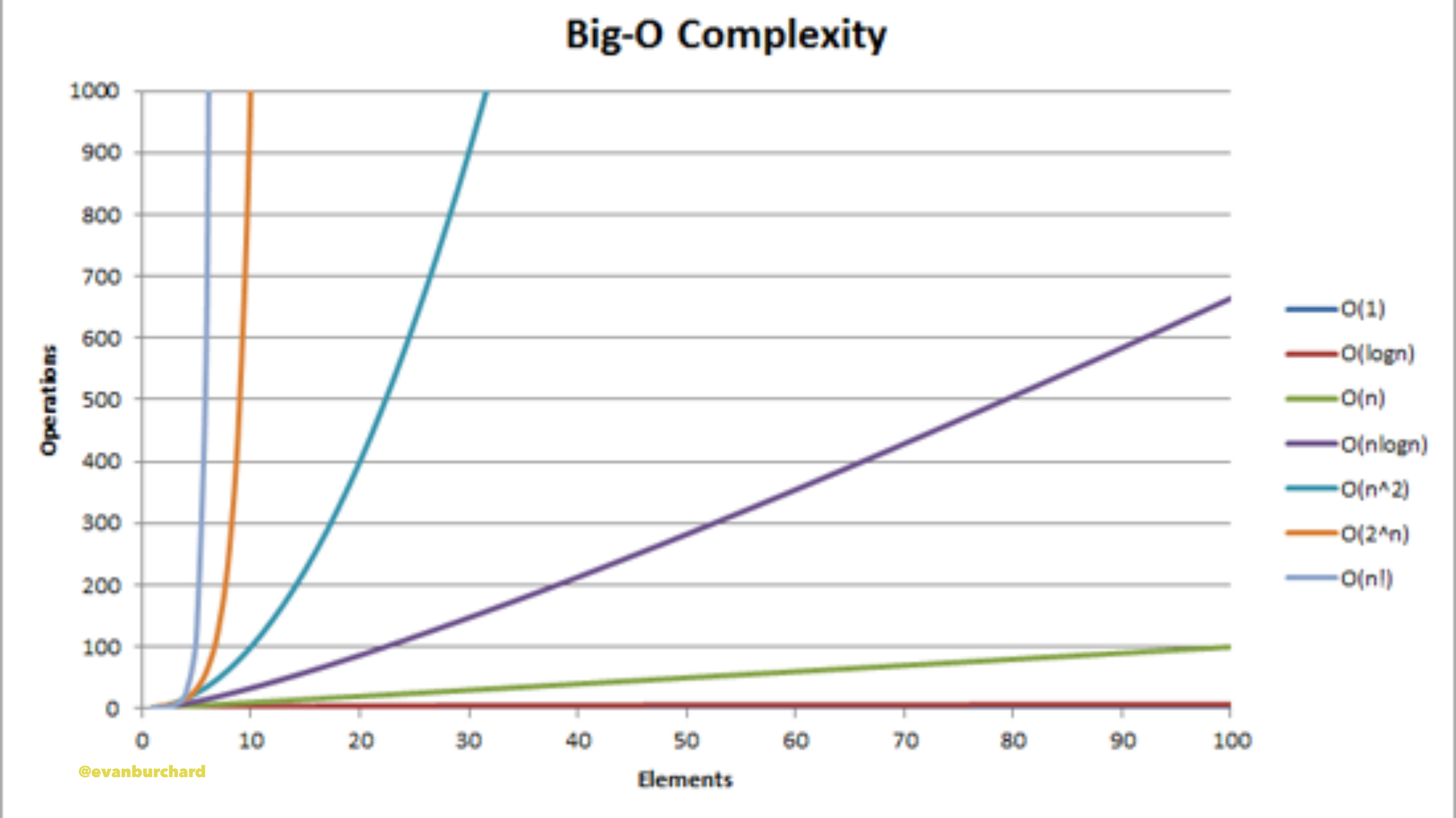
- Time/Space Complexity
- Analytic Geometry
- Functions

Elements of Big O

- Time/Space Complexity
- Analytic Geometry
- Functions
- Inputs and Outputs

Time/Space Complexity classes

- O(1)
- O(log(n))
- O(n)
- O(n^2)


Big-O Complexity
Operations
1000
900
800
700
600
500
400
300
200
100
0
0
10
20
30
40
50
60
70
80
90
100
Elements
O(1)
O(logn)
O(n)
O(nlogn)
O(n^2)
O(2^n)
O(n!)
@evanburchard

Time/Space Complexity classes

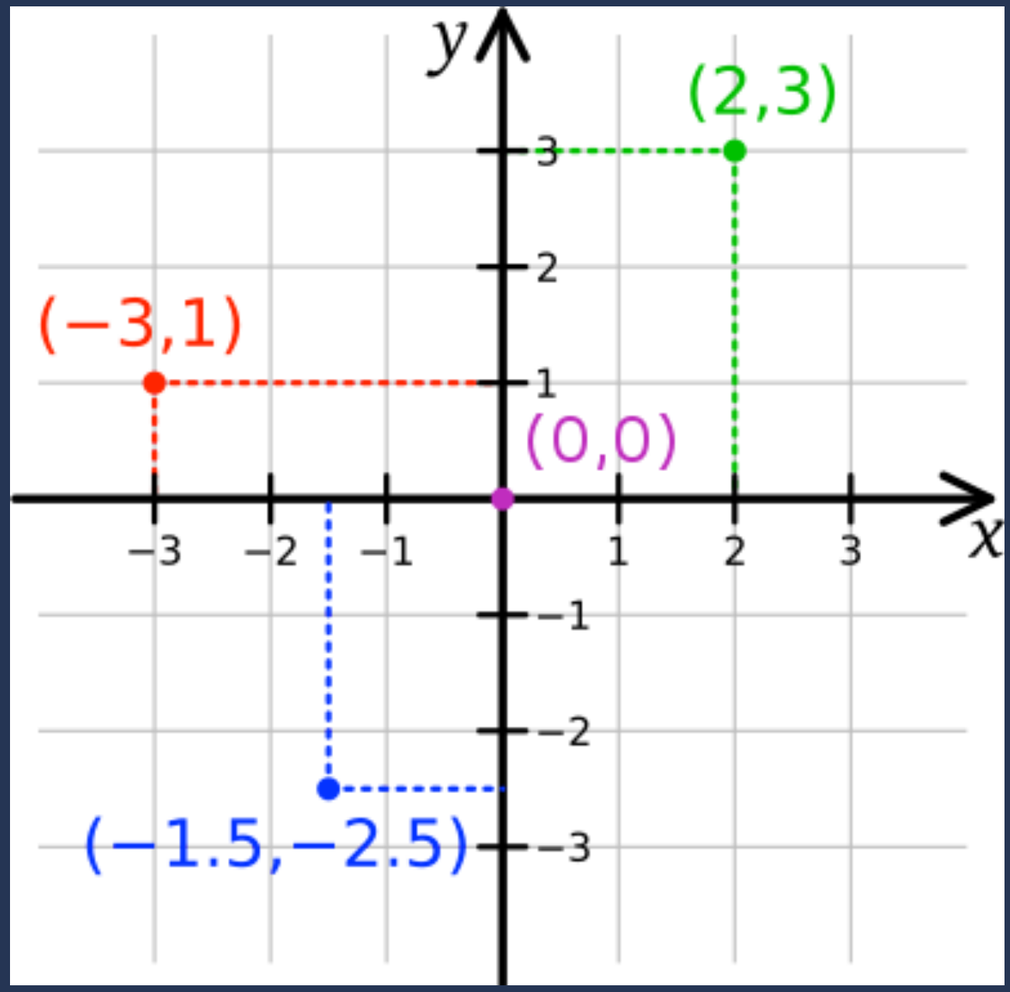
The reason this is so heavily emphasized is because so many of the techniques we use, regardless of the structure or theory underneath is made so that our performance is O(1) whenever possible.

Analytic Geometry

You might have learned this around the same time as algebra. It's the one where you make graphs.

If you're looking at big O notation, a refresher on this can help you contextualize your work.

Analytic Geometry



Functions

```
f(x) = x + 2
```

Functions

```
f(x) = x + 2
```

You could also write it as:

Functions

```
f(x) = x + 2
```

You could also write it as:

```
y = x + 2
```

Functions

```
f(x) = x + 2
```

You could also write it as:

```
y = x + 2
```

Both demonstrate different things.

functions

```
y = x + 2
```

y is the "dependent" variable, and x is the "independent" variable.

functions

```
y = x + 2
```

y is the "dependent" variable, and x is the "independent" variable.

y can also be said to be a "function of x."

functions

```
f(x) = x + 2
y = x + 2
```

Both forms can be combined like this:

```
y = f(x) = x + 2
```

functions

```
y = f(x) = x + 2
```

functions

```
y = f(x) = x + 2
```

This probably looks like something you know.

functions

```
y = f(x) = x + 2
```

This probably looks like something you know.

```
y = (x) => x + 2
y = -> (x) {x + 1}
```

functions

```
y = f(x) = x + 2
```

This probably looks like something you know.

```
y = (x) => x + 2
y = -> (x) {x + 1}
```

Q: What is the first one?

functions

```
y = f(x) = x + 2
```

This probably looks like something you know.

```
y = (x) => x + 2
y = -> (x) {x + 1}
```

Q: What is the second one?

functions

`y = f(x) = x + 2` (function)

`y = (x) => x + 2` (ES6 Syntax)

`y = -> (x) {x + 1}` ("stabby proc" syntax)

Functions

- f(x) = x + 2
- y = x + 2

Q: What is the big O(x) of these functions? (note that "O(n)" is more conventional)?

Functions

A linear function O(n), will use do some processing for each input, so

```
[3, 4, 5].map{ |n| n + 3 }
# or
[3, 4, 5].map(&plus_3) # from earlier
# or
[3, 4, 5].inject(:+)
```

Inputs and Outputs

Inputs and Outputs

- y = x + 2

Inputs and Outputs

- y = x + 2
- Q: What are the inputs and outputs of this function?

Inputs and Outputs

- y = x + 2
- Q: What are the inputs and outputs of this function?
- Q: How would we write it in Ruby?

Inputs and Outputs

- y = x + 2
- Q: What are the inputs and outputs of this function?
- Q: How would we write it in Ruby?

```
def add_two(x)
  x + 2
end
```

Inputs and Outputs

- y = x + 2
- Q: What are the inputs and outputs of this function?
- Q: How would we write it in Ruby?
- Q: How is this different?

```
def add_two(x)
  puts x + 2
end
```

functions vs. methods vs. procedures

An argument against algorithmic emphasis

"It's not like we're building our own data structures"

Hold up

Binary Search all the time

Binary Search all the time

- Q: Git Bisect

Binary Search all the time

- Git Bisect
- Searching through a file or log output

Decision Trees informally all the time

- Selecting packages

Decision Trees informally all the time

- Selecting packages
- Trying out solutions

In other words

Optimization all the time

Optimization all the time

- Git Bisect

Optimization all the time

- Git Bisect
- Searching through a file or log output

Optimization all the time

- Git Bisect
- Searching through a file or log output
- Selecting packages

Optimization all the time

- Git Bisect
- Searching through a file or log output
- Selecting packages
- Trying out solutions

And also

Complexity collapsing techniques like

Complexity collapsing techniques like

- Caching

Complexity collapsing techniques like

- Caching
- Memoization

Complexity collapsing techniques like

- Caching
- Memoization
- Closed Form Functions

Complexity collapsing techniques like

- Caching
- Memoization
- Closed Form Functions
- Presorting

Complexity collapsing techniques like

- Caching
- Memoization
- Closed Form Functions
- Presorting
- Indexing

Complexity collapsing: Caching

Complexity collapsing: Caching

In Rails this could mean a lot of different things: page caching, fragment caching, query caching.

Complexity collapsing: Memoization

Complexity collapsing: Memoization

For nullary (arity of 0) functions:

```
def memoized_function
  @var ||= some_calculation
end
```

Complexity collapsing: Memoization

For nullary (arity of 0) functions:

```
def memoized_function
  @var ||= some_calculation
end
```

Further discussion on the Memoization deprecation in Rails is here:
https://github.com/rails/rails/commit/36253916b0b788d6ded56669d37c96ed05c92c5c

Complexity collapsing: Memoization

Also, "materialized views" are this technique for databases

Complexity collapsing: Memoization

Last thing on memoization... There are certain ways to write our code to make them always memoizable. They involve functional programming.

Complexity collapsing: Memoization

Basically any function that receives the same input should produce the same output. This is a "pure function" with the property of "referential transparency."

Complexity collapsing: Memoization

Basically any function that receives the same input should produce the same output. This is a "pure function" with the property of "referential transparency."

This means functions can be turned into values, and "calling" them is just a O(1) lookup.

Referential Transparency: Limitations

- Global variables (evil, right?)

Referential Transparency: Limitations

- Global variables (evil, right?)
- Implicit parameters, aka, receivers of a method, aka, objects! (evil too, really?)

Referential Transparency: Limitations

- Global variables (evil, right?)
- Implicit parameters, aka, receivers of a method, aka, objects! (evil too, really?)
- Your database, yep. It's just a giant evil global variable.

Complexity collapsing: Closed Form Functions

Complexity collapsing: Closed Form Functions

Complexity collapsing: Closed Form Functions

Q: What is the closed form of this?

Complexity collapsing: Closed Form Functions

$$\sum_{i=0}^{n} i = = \frac{n(n+1)}{2}$$

Complexity collapsing: Closed Form Functions

$$\sum_{i=0}^{n} i = \frac{n(n+1)}{2}$$

Q: What about in Ruby?

Complexity collapsing: Closed Form Functions

$$\sum_{i=0}^{n} i = \frac{n(n+1)}{2}$$

Q: What's the advantage of one compared to the other?

Complexity collapsing: Closed Form Functions

By the way, "Concrete Mathematics," which is kind of a primer to "The Art of Programming" spills a ton of ink on this subject.

Complexity collapsing: Presorting/Indexing

Complexity collapsing: Presorting/Indexing

```
class AddIndexToTable < ActiveRecord::Migration
  def change
    add_index :table, :column_to_index
  end
end
```

Complexity collapsing: Presorting/Indexing

If you want to get started with indexing, try the "lol dba" gem.

It will tell you where you're missing indexes.

Complexity collapsing: Presorting/Indexing

If you want to get started with indexing, try the "lol dba" gem.

It will tell you where you're missing indexes.

Spoiler, you're probably missing them on foreign keys.

Complexity collapsing: Presorting/Indexing

If you want to get started with indexing, try the "lol dba" gem.

It will tell you where you're missing indexes.

Spoiler, you're probably missing them on foreign keys.

Also, it's good to note that you already have them on primary keys.

Complexity collapsing: Presorting/Indexing

Q: Indexing Tradeoffs?

Complexity collapsing

One last thing on making things more efficient. It's a business imperative, yes. But there's something fundamentally interesting about it too.

- Minecraft
- Clicker Games
- Discovery of actually faster algorithms

So math helps us optimize our code, but we need to be able to read it too.

Syntax

Syntax

Q: What math syntax might we use for a common O(n) (linear) function?

Syntax

Q: What math syntax might we use for a common O(n) (linear) function?

How about this?

$$\sum_{i=0}^{n} i$$

Q: What the hell is that Pointy E?

Q: What the hell is that Pointy E?

That's a "sigma," from Greek, and it denotes summation notation. If you're looking at math stuff and it doesn't make sense, you might need to find the symbol first.

Q: What the hell is that Pointy E?

That's a "sigma," from Greek, and it denotes summation notation. If you're looking at math stuff and it doesn't make sense, you might need to find the symbol first.

(It might a greek letter, lowercase or uppercase)

Six problems with math

1. What is this particular thing called?
2. What does the arrangement of these things mean?
3. What is this whole structure called?
4. What is the actual answer? I can't run this.
5. All of the variable names are single letters!
6. I can't copy and paste!

Happens in programming too.

A few I had a hard time with:

- Ternary Syntax

  ```
  a ? b : c
  ```

- here document (heredoc) strings
- metasyntactic variables (aka "foo" & friends)

Happens in programming too.

A few I had a hard time with:

- Ternary Syntax

  ```
  a ? b : c
  ```

- here document (heredoc) strings
- metasyntactic variables (aka "foo" & friends)

Q: Anyone else have a thing they found hard to look up and learn the name for?


3
i
i=1

Q: What the hell is that Roman Numeral two with a hat?

$$\prod_{i=1}^{3} i$$

What the hell is that Roman Numeral two with a hat?

π.upcase = Π

It's the product notation, like the sigma sum.

π.upcase = $\prod$

Q: So what is this?

$$\prod_{i=1}^{3} i$$

So what is this?

It's the factorial!

$$\prod_{i=1}^{3} i = 3!$$

So what does this equal?

$$\prod_{i=1}^{3} i = 3! = 6$$

Q: What is that in Ruby?

$$\prod_{i=1}^{3} i = 3!$$

Q: What is that in Ruby?

$$\prod_{i=1}^{3} i = 3!$$

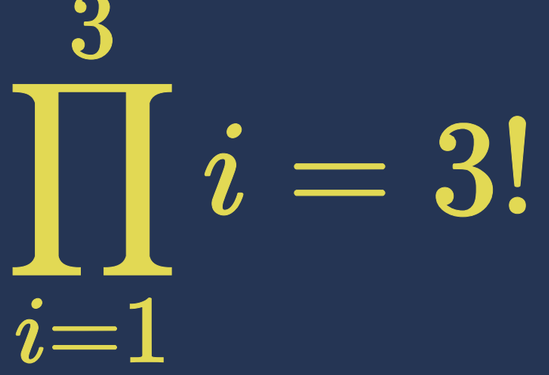
3!

Q: What is that in Ruby?

$$\prod_{i=1}^{3} i = 3!$$

3! - NOPE

Q: What is that in Ruby?



`3!` - NOPE

```
Math.factorial(3)
```

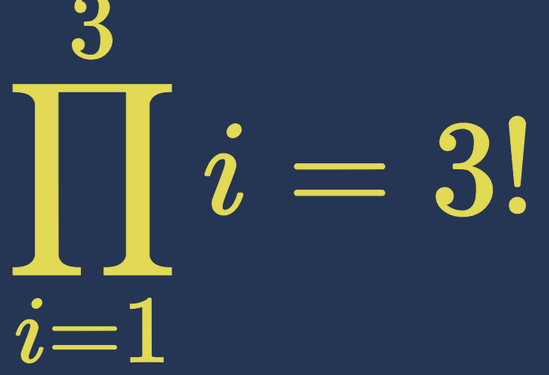
Q: What is that in Ruby?

$$\prod_{i=1}^{3} i = 3!$$

`3!` - NOPE

`Math.factorial(3)` - NOPE

Q: What is that in Ruby?

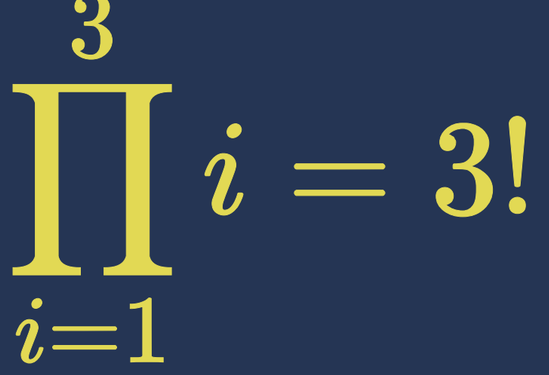


`3!` - NOPE

`Math.factorial(3)` - NOPE

`Math.fact(3)`

Q: What is that in Ruby?

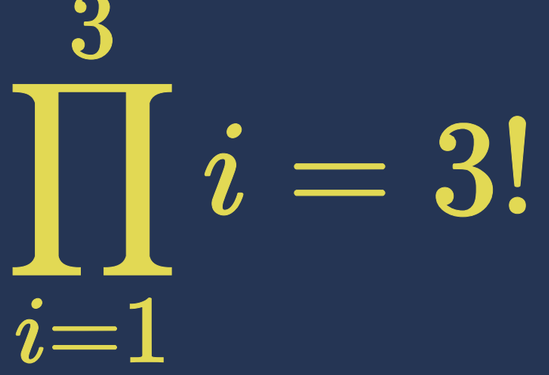


`3!` - NOPE

`Math.factorial(3)` - NOPE

`Math.fact(3)` - NOPE

Q: What is that in Ruby?

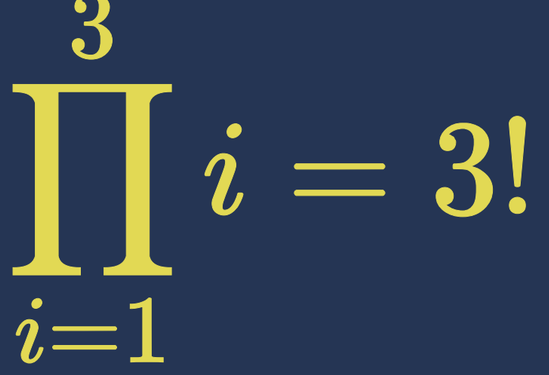


`3!` - NOPE

`Math.factorial(3)` - NOPE

`Math.fact(3)` - NOPE

Q: What is that in Ruby?



`3!` - NOPE

`Math.factorial(3)` - NOPE

`Math.fact(3)` - NOPE

Q: What is that in Ruby?

$$\prod_{i=1}^{3} i = 3!$$

`3!` - NOPE

`Math.factorial(3)` - NOPE

`Math.fact(3)` - NOPE

`Math.gamma(3)` - NOPE

`Math.gamma(4)`

Q: What is that in Ruby?

$$\prod_{i=1}^{3} i = 3!$$

`3!` - NOPE

`Math.factorial(3)` - NOPE

`Math.fact(3)` - NOPE

`Math.gamma(3)` - NOPE

`Math.gamma(4)` - OK... kinda

Q: What is that in Ruby?

$$\prod_{i=1}^{3} i = 3!$$

`3!` - NOPE

`Math.factorial(3)` - NOPE

`Math.fact(3)` - NOPE

`Math.gamma(3)` - NOPE

`Math.gamma(4)` - OK... kinda

`(1..3).inject(:*)`

Q: What is that in Ruby?

$$\prod_{i=1}^{3} i = 3!$$

`3!` - NOPE

`Math.factorial(3)` - NOPE

`Math.fact(3)` - NOPE

`Math.gamma(3)` - NOPE

`Math.gamma(4)` - OK... kinda

`(1..3).inject(:*)` - Perfect

That's a bit unexpected.

That's a bit unexpected.

In Ruby's math, I guess we can't always count on the principle of least surprise.

Expected Value

Expected Value

The value you expect, given a set of known probabilities and values.

Expected Value

$$E(X) = x_1p_1 + x_2p_2 + x_3p_3 + \ldots + x_np_n$$

Expected Value

$$E(X) = x_1p_1 + x_2p_2 + x_3p_3 + \ldots + x_np_n$$

also

$$E(X) = \sum x_i f(x_i)$$

Expected Value

$$E(X) = x_1p_1 + x_2p_2 + x_3p_3 + \ldots + x_np_n$$

$$E(X) = \sum x_i f(x_i)$$

Q: These are the same, right?

Expected Value

How can we use this?

Expected Value

How can we use this?

- Poker

Expected Value

How can we use this?

- Poker
- Evaluating Packages

Expected Value

How can we use this?

- Poker
- Evaluating Packages
- Evaluating Jobs

An example:

$$E(X) = \sum x_i f(x_i)$$

```
y_companator = [{salary: 90, equity: 0.10, final_valuation: 2_000_000_000, odds: 0.005},
                {salary: 90, equity: 0.10, final_valuation: 0, odds: 0.995}]

better_safe_and_salary_llc = {salary: 120}
```

An example:

$$E(X) = \sum x_i f(x_i)$$

```
y_companator = [{salary: 90, equity: 0.10, final_valuation: 2_000_000_000, odds: 0.005},
                {salary: 90, equity: 0.10, final_valuation: 0, odds: 0.995}]

better_safe_and_salary_llc = {salary: 120}
```

Q: Anyone have it?

An example:

Q: Anyone have it?
(hint, the expected value of that 10% equity is 1 million dollars)

But what about low exits? What about less equity? What about with 4 years until IPO?

salary = 4 * 90
(with odds of 0.0001 of the big (2 billion) exit, only 0.0001 equity)

```
y_companator = [{salary: 90, equity: 0.0001, final_valuation: 2_000_000_000, odds: 0.0001},
                {salary: 90, equity: 0.0001, final_valuation: 0, odds: 0.9999}]

better_safe_and_salary_llc = {salary: 120}
```

Guesses?

But what about low exits? What about less equity? What about 4 years till IPO?

But what about low exits? What about less equity? What about 4 years till IPO?

Then the equity is worth $20

But what about low exits? What about less equity? What about 4 years till IPO?

Then the equity is worth $20

And our llc worker is up $119980 after 4 years.

Expected Value

If you want to do this for real, we would model a few exits (with the same company) and sum their probabilities. We did an easy version with only one outcome with a value and probability.

Expected Value

In poker, this makes way more sense because the probabilities are based on realities, and you get to play a lot more hands.

Expected Value

But also, don't use this for planning (or probably poker either). Use this maybe in a negotiation or for fun day dreaming.*

Expected Value

But also, don't use this for planning (or probably poker either). Use this maybe in a negotiation or for fun day dreaming.*

* This is not financial advice.

Let's play monopoly

The Herfindahl Index - Determines whether a market is monopolized

$$\sum_{i=1}^{N} {s_i}^2$$

$$\sum_{i=1}^{N} {s_i}^2$$

N = The number of firms

i = The index of the firm (1..N)

S_i = The market share of firm i in the market

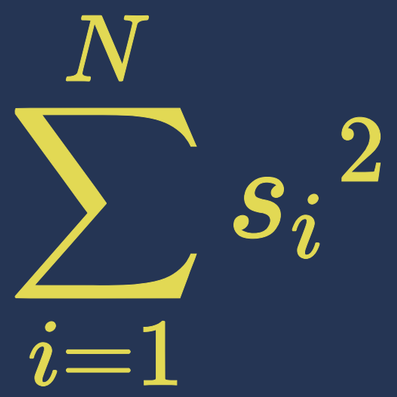
$$\sum_{i=1}^{N} {s_i}^2$$

N = The number of firms

i = The index of the firm (1..N)

S_i = The market share of firm i in the market

Q: What about in Ruby?




```
firms.inject{|sum, firm| firm.market_share^2 += sum}
```

$$\sum_{i=1}^{N} {s_i}^2$$

```
firms.inject{|sum, firm| firm.market_share^2 += sum}
```

Q: How do we get there?

Herfindahl Index

- An H below 0.01 (or 100) indicates a highly competitive industry.

Herfindahl Index

- An H below 0.01 (or 100) indicates a highly competitive industry.
- An H below 0.15 (or 1,500) indicates an unconcentrated industry.

Herfindahl Index

- An H below 0.01 (or 100) indicates a highly competitive industry.
- An H below 0.15 (or 1,500) indicates an unconcentrated industry.
- An H between 0.15 to 0.25 (or 1,500 to 2,500) indicates moderate concentration.

Herfindahl Index

- An H below 0.01 (or 100) indicates a highly competitive industry.
- An H below 0.15 (or 1,500) indicates an unconcentrated industry.
- An H between 0.15 to 0.25 (or 1,500 to 2,500) indicates moderate concentration.
- An H above 0.25 (above 2,500) indicates high

Herfindahl Index

Note, this is also called the "Simpson Index" and you can use it to measure diversity in populations.

Gini Coefficient of Wealth Inequality

@evanburchard

Gini Coefficient of Wealth Inequality

$$G = \frac{1}{2\mu} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} p(x)p(y) \left| x - y \right| dxdy$$

Gini Coefficient of Wealth Inequality

$$G = \frac{1}{2\mu} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} p(x)p(y) \left| x - y \right| dxdy$$

jk...

$$G = \frac{1}{2\mu} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} p(x)p(y) \left| x - y \right| dxdy$$

Worth noting though that some of this should be a little more decipherable than before.

* The μ is a "mu," a Greek letter, and if we looked that up, we'd see it meant "mean".
* We could probably expand some variables, but really here, we're stuck with calculus... and we're in Ruby.

Another thing to note

With some calculations, Integrals become Summations, which is also the difference between "continuous" and "discrete" ranges of values.

PAPERS

Incentivizing the Dissemination of Truth...

https://arxiv.org/pdf/1804.02509

One of the best and worst things I've seen in a math/cs paper...

Table 1. Variables and Description

Variable	***Table column subhead***
N	Number of regular users that could potentially disemminate truth in the network
M	Number of volunteering agents among regular agents
c	Cost of volunteering for regular agents
α	Cost of failing to dominate the news cycle
k	Minimum group size to dominate the news cycle: the number of volunteering fake news agents
x	Volunteering ratio and the probability of volunteering in symetric model
σ	Aggregated reward shared among regular agents
$p_v(M)$ / $p_v(k)$	Payoff of volunteering for a regular/fake news agent as function of the number of volunteers
$p_d(M)$ / $p_d(k)$	Payoff of defecting for a regular/fake news agent as function of the number of volunteers
$\overline{P_v}(k)$ / $\overline{P_d}(k)$	Average payoff of volunteering/defecting regular agents as function of group size
F	Number of fake news users that could potentially diseminate fake news
c_f	Cost of volunteering for fake news agents
$\overline{P_v}(M)$ /$\overline{P_d}(M)$	Average payoff of volunteering/defecting fake news agents as a function of regular volunteering agents
P*	Equilibrium probability of volunteering for regular agents
$\overline{P_v}$ / $\overline{P_d}$	Expected value of volunteering/defecting payoff of fake news agents

Incentivizing the Dissemination of Truth...

$$p_v(M) = \begin{cases} 1 - c, & \text{if } \mathrm{M} \geq \mathrm{k} \\ 1 - a - c, & \text{if } \mathrm{M} < \mathrm{k} \end{cases}$$

Incentivizing the Dissemination of Truth...

$$p_v(M) = \begin{cases} 1 - c, & \text{if M} \geq \text{k} \\ 1 - a - c, & \text{if M} < \text{k} \end{cases}$$

Q: How would we rewrite this in Ruby?

Learned Index Structures

Learned Index Structures

Goodbye B-Trees?

https://arxiv.org/abs/1712.01208v2

Learned Index Structures

Goodbye B-Trees?

https://arxiv.org/abs/1712.01208v2

$$D = (x_i, y_i = 1)|x_i \in K \cup (x_i, y_i = 0)|x_i \in U$$

Learned Index Structures

Goodbye B-Trees?

https://arxiv.org/abs/1712.01208v2

$$D = (x_i, y_i = 1)|x_i \in K \cup (x_i, y_i = 0)|x_i \in U$$

This uses "set builder notation."

Set Builder Notation in Wikipedia

	Example 1	Example 2
Set-builder	$\{l \mid l \in L\}$	$\{(k, x) \mid k \in K \land x \in X \land P(x)\}$
Python	`{l for l in L}`	`{(k, x) for k in K for x in X if P(x)}`
Haskell	`[l | l <- ls]`	`[(k, x) | k <- ks, x <- xs, p x]`
Scala	`for (l <- L) yield l`	`for (k <- K; x <- X if P(x)) yield (k,x)`

@evanburchard

What about Ruby?

```
L.map{|1| 1}
```

What about Ruby?

```
L.map{|1| 1}
```

Not exactly the same. We kind of want just the block.

Not exactly the same, but with enumerable:

- select (find_all)
- inject (reduce)
- reject
- detect (find)
- collect (map)

By the way, Map/Reduce is this...

map

reduce

But back to set theory...

A handful of set theory symbols

∈: is an element of

∪: union

∩: intersection

∧: and

∨: or

⊆: is a subset of

⊇: is a superset of

⊃: is a (proper) subset of

⊂: is a (proper) superset of

@evanburchard

A handful of set theory symbols

|: "such that"

{}: defines a set

¬: not

∀: for all

∃: there exists

https://en.wikipedia.org/wiki/Glossary_of_set_theory

Closing tips

- Some things are complex. Others are just unfamiliar.
- Spending time in other programming environments can help your Ruby.
- If you see some math you don't understand, see how far you can get by unwinding it into Ruby.
- Try to find sample code, and maybe write your own version.
- The math reference might still be useful.

Questions?

@evanburchard

Resources

- MathJax https://docs.mathjax.org/en/latest/
- MathJax Reference https://math.meta.stackexchange.com/questions/5020/mathjax-basic-tutorial-and-quick-reference

Resources: Complexity Collapsing

- http://guides.rubyonrails.org/caching_with_rails.html
- https://github.com/rails/rails/commit/36253916b0b788d6ded56669d37c96ed05c92c5c
- Sums, Products & Asymptotics 1 Closed Forms and Approximations https://dspace.mit.edu/bitstream/handle/

Resources: Complexity Collapsing

- "Concrete Mathematics" for Closed Form-ifying
- Indexing: https://github.com/plentz/lol_dba

Resources: Monopoly

- https://en.wikipedia.org/wiki/Herfindahl_index
- https://en.wikipedia.org/wiki/Gini_coefficient

Papers:

Learned index Structures
- https://arxiv.org/abs/1712.01208v2

Incentivizing the Dissemination of Truth...
- https://arxiv.org/abs/1804.02509

Another cool paper I couldn't include :(
- https://arxiv.org/abs/1802.00121

More references

- https://www.arxiv-vanity.com/papers/
- http://paperswelove.org/
- https://en.wikipedia.org/wiki/List_of_mathematical_symbols
- https://en.wikipedia.org/wiki/Glossary_of_set_theory
- https://en.wikipedia.org/wiki/Set-builder_notation

Too much calculus
- https://math.temple.edu/~siagla/articles/Strang2001.pdf

Maybe you just want some recreational programming

- Metaprogramming
- Obfuscated Code
- Code Golf
- "Programming With Nothing" - Church Numerals, Combinators, (untyped) Lambda Calculus https://codon.com/programming-with-nothing

More recreational programming

- Project Euler (These problems are especially good at pointing out Ruby's pros and cons)
- Destroy All Software's projects
- Nand2Tetris http://www.nand2tetris.org/
- Lambda Calculus - Fundamentals of Lambda Calculus & Functional Programming in JavaScript https://www.youtube.com/watch?v=3VQ382QG-y4

Tropes

http://tvtropes.org/pmwiki/pmwiki.php/Main/OneLastJob
http://tvtropes.org/pmwiki/pmwiki.php/Main/LongGame

That's it

Thanks.

@evanburchard

Me Again

- Evan Burchard: @evanburchard on Twitter/GitHub
- This talk: github.com/evanburchard/math*for*rails
- Web Game Developer's Cookbook: jsarcade.com (code)
- Refactoring JavaScript: refactoringjs.com (full book as PDF)
- Job hunting site: jobpika.com