Who is the Mightiest Avenger?

Matthew Roughan
https://aleph-zero-heros.info/
Applied probability and the importance of programming

Computing is integral to modern mathematics

- Core of industrial work is implementation

- Many AP problems can be attacked more quickly (and sometimes more realistically) via simulation

- Programming makes ideas real, and imposes discipline on work

And we are fighting a rear-guard action to be part of data science

- Too often DS is portrayed as “model free”
Julia is a (newish) open-source programming language

- Syntax is reminiscent of Matlab
- Purpose is similar to Matlab, R, Python
  - Numerical computing
  - Data science
- Challenges standard intuition about programming paradigms
  - High-performance
  - Without sacrificing productivity

“Julia walks like Python and runs like C”
Julia 1.0

- Julia 1.0 released August 2018 (and 1.1 in May 2019)
  - Stable now, at last!
  - Performance increase from 0.6 -> 1.0
    - 71% time taken for large (surreal) multiplication
    - Performance has always been a crucial motivation for Julia

- I am now using Julia almost exclusively
  - Replaced Matlab (except for teaching)
  - Replaced Perl (except for legacy code)
  - Replaced C
  - Replacing Python (except for certain packages)

- It’s not perfect, but it is getting better
  - Biggest limits are in the community size / package availability
Why Julia (over Matlab)

- Open Source
- Modern data structures and idioms
  - Comprehensions
  - Dictionaries (native)
  - Data Frames (package)
  - Don’t need to vectorise everything
- Data Types and Multiple Dispatch
  - Example: Distributions package
    
    ```julia
    julia> D = Normal(1.0)
        julia> quantile.(D, [0.5, 0.95])
    Returns [1.0, 2.64]
    ```
- But it isn’t OO (Object Oriented)
Learn by doing

- You don’t learn programming languages by reading

- I needed a good project:
  - Simple(-ish)
  - Adds value to Julia community
  - But isn’t in the critical pathway, because I am not that good a Julia coder yet

- So
  - Package to do ratings
  - Analysis of a new dataset
(Sports) Ratings

- A common aim is to rate teams or players
  - Should be related to strength (not just a ranking)
  - Should be objective, based on performance
  - Should take into account the strength of opponents

- There are many systems
  - Linear algebra
    - Massey
    - Colley
    - Eigen-rating
  - Probabilistic
    - Elo
    - Glicko
Elo Ratings

- Invented by Arpad Elo for rating Chess players

- Elo presumes that the outcome of a contest is random
  - Each player has a strength $S_i$
  - Player performance is a RV with location parameter $= S_i$
  - Elo initially used Normal, then Gumbel
  - Winner has higher performance (on the day)

- Ratings $R_i$ are estimates of $S_i$
  - Elo adopted recursive/iterative update rule
  - Player A v B, outcome is $O_{AB}$ and expected outcome is $E_{AB}$
    - Outcome probability assumed to be Logistic*

$$R_A' = R_A + K(O_{AB} - E_{AB})$$

* The difference of 2 Gumbels is Logistic
There are many ratings schemes and variants

Elo, for instance, has issues
  - Inflation
  - Two-player not multi-
  - Parameter choice (e.g., K)

Most software for using Elo is only that
  - I want to be able to experiment

I wanted an extensible, mathematically sound package

Julia is ideal
  - Numerically fast
  - Good use of memory
  - Easy to program
  - Supportive data structures
Ratings Package in Julia

RatPack.jl

- Simple package design
  - Add new ratings system by adding
    - Definition source code
    - Type (for passing parameters)
    - Calculation rule (map outcomes to ratings)
  - A hook in the main package
- Includes wrappers
  - e.g., iterate over an update rule
  - e.g., simulation of tournaments to feed into tests
  - e.g., scoring and cross-validation

- It isn’t quite complete yet, and needs better docs, but I need some nice test cases to play with first
The Marvel Cinematic Universe (the MCU)

- Most successful film franchise ever

<table>
<thead>
<tr>
<th>Franchise</th>
<th>Start year</th>
<th># movies</th>
<th>Box office (US) *</th>
</tr>
</thead>
<tbody>
<tr>
<td>MCU</td>
<td>2008</td>
<td>23</td>
<td>US$ 8.7 billion</td>
</tr>
<tr>
<td>Star Wars</td>
<td>1977</td>
<td>11</td>
<td>US$ 7.5 billion</td>
</tr>
<tr>
<td>James Bond</td>
<td>1963</td>
<td>25</td>
<td>US$ 5.8 billion</td>
</tr>
<tr>
<td>Harry Potter</td>
<td>2001</td>
<td>10</td>
<td>US$ 3.6 billion</td>
</tr>
<tr>
<td>Batman</td>
<td>1989</td>
<td>9</td>
<td>US$ 2.9 billion</td>
</tr>
</tbody>
</table>

https://www.the-numbers.com/movies/franchises/sort/World

* inflation adjusted
MCU and a theory of conflict

- Different genres have different drivers
  - Musicals are driven by songs
  - Dramas are driven by dialogue (most network analysis)
  - Action movies are driven by action
    - A large component of action is conflict

- Watched and annotated the entire MCU
  - Data is a set of rows, 1 per “conflict”
    - Conflicts broken down as much as possible to max data
    - Each row has time, nature of conflict, parties involved + winner (or inconclusive), and factors affecting outcome

- Simple (seeming) question: who is mightiest?
### Example Data: Captain Marvel

<table>
<thead>
<tr>
<th>Time</th>
<th>Type of conflict</th>
<th>Party A</th>
<th>$F_A$</th>
<th>Party B</th>
<th>$F_B$</th>
<th>Outcome</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.32</td>
<td></td>
<td>Captain Marvel</td>
<td>-2</td>
<td>Yon-Rogg</td>
<td></td>
<td>Yon-Rogg</td>
</tr>
<tr>
<td>2.46</td>
<td></td>
<td>Captain Marvel</td>
<td>-2</td>
<td>Yon-Rogg</td>
<td></td>
<td>inconclusive</td>
</tr>
<tr>
<td>2.53</td>
<td></td>
<td>Captain Marvel</td>
<td>-2</td>
<td>Yon-Rogg</td>
<td></td>
<td>Yon-Rogg</td>
</tr>
<tr>
<td>3.02</td>
<td></td>
<td>Captain Marvel</td>
<td>-2</td>
<td>Yon-Rogg</td>
<td></td>
<td>Yon-Rogg</td>
</tr>
<tr>
<td>3.25</td>
<td></td>
<td>Captain Marvel</td>
<td>-2</td>
<td>Yon-Rogg</td>
<td></td>
<td>Yon-Rogg</td>
</tr>
<tr>
<td>3.31</td>
<td></td>
<td>Captain Marvel</td>
<td>-2</td>
<td>Yon-Rogg</td>
<td></td>
<td>Captain Marvel</td>
</tr>
<tr>
<td>6.22</td>
<td>mental</td>
<td>Captain Marvel</td>
<td></td>
<td>Supreme Intelligence</td>
<td>Supreme Intelligence</td>
<td></td>
</tr>
</tbody>
</table>
Conflict Graph (physical)

Nodes:
- Captain Marvel
- Minn-Erva
- Bron-Char
- Norex
- Att-Lass
- Yon-Rogg
- Talos
- Korath
- Mar-Vell
- Nick Fury

Edges with labels:
- Minn-Erva to Captain Marvel: 5
- Bron-Char to Captain Marvel: 8
- Norex to Captain Marvel: 9
- Att-Lass to Captain Marvel: 3
- Yon-Rogg to Captain Marvel: 14
- Talos to Captain Marvel: 7
- Korath to Captain Marvel: 5
- Mar-Vell to Captain Marvel: 1
- Nick Fury to Captain Marvel: 4

Weights:
- Minn-Erva to Captain Marvel: 5
- Bron-Char to Captain Marvel: 8
- Norex to Captain Marvel: 9
- Att-Lass to Captain Marvel: 3
- Yon-Rogg to Captain Marvel: 14
- Talos to Captain Marvel: 7
- Korath to Captain Marvel: 5
- Mar-Vell to Captain Marvel: 1
- Nick Fury to Captain Marvel: 4
Results

- MCU conflict data
  - average = 1500
  - human < 1250
  - “heroic” 1250-1750
  - “super” > 1750
  - “godlike” > 2000

- Sampled iteration
  - Resample conflicts
  - Reduce K over time

- Not bad
  - Some anomalies
  - Doesn’t account for resilience
  - Doesn’t account for “team-up” effect
Conclusion

- Julia
  - Great new programming language
  - Should replace many older tools

- RatPack.jl
  - Ratings test package as a way to test Julia and learn more

- MCU research is just starting, for some other bits see
  
  https://aleph-zero-heros.info/
  @AlephZeroHeros