Ranking Ultimate Frisbee Teams

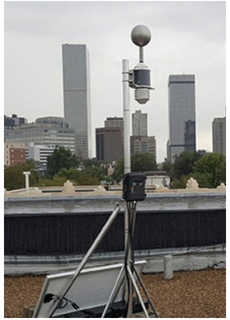
Insight Bayesian Workshop

Shane Bussmann
July 18, 2018

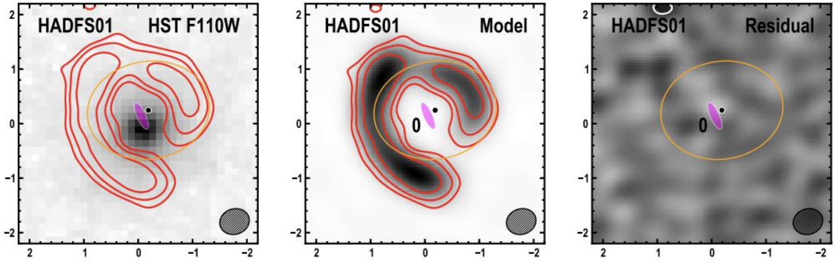
About Me

- Senior Data Scientist at CiBO Technologies since December 2017
- Lead Data Scientist at Understory Weather, 2015 - 2017
- Health Data Science Fellow, Boston, Summer 2015
- Academic history
  - PhD Astronomy, 2010, University of Arizona
  - Postdoc at Harvard-Smithsonian CfA, 2010 - 2013
  - Postdoc at Cornell, 2013 - 2015

Understory



Astronomy

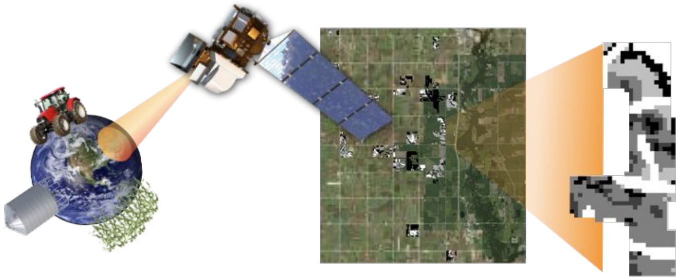


CiBO Technologies is

Planetary-Scale

Agricultural

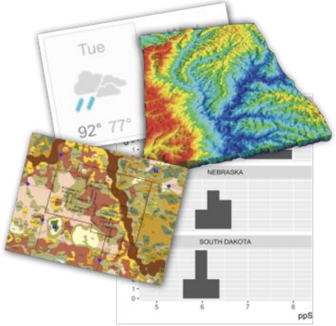
Simulation Optimization



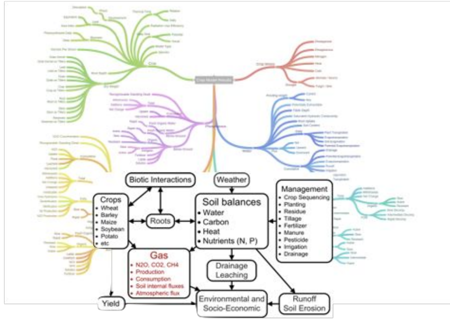
Planetary Scale, Daily Simulations

Sub-field resolution utilizing on-farm and remote sensing

Multi-country, Multi-crop



Thousands of Variables
&
Millions of data points *per field*



Rich Domain Model with Hundreds of Interactions and Outputs

Goal of This Workshop

Walk through a real-world example to show how Bayesian analysis leads to better results!

The Problem

- BUDA (Boston Ultimate Disc Alliance) runs recreational ultimate frisbee leagues in the greater Boston area
- We have been asked to rank teams for the end-of-season tournament
- Teams self-assign to one of three divisions
  - Div 1 == highest
  - Div 3 == lowest
- Teams responsible for creating their own schedule
  - Some teams play lots of games, others not so much
  - Some teams play “out of division” games (e.g., Div 1 vs. Div 2)

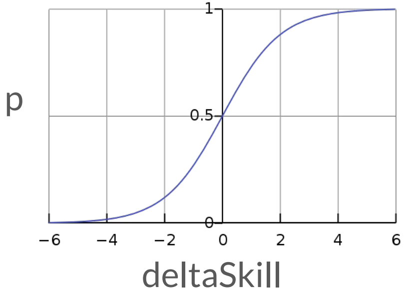
Ranking Teams: Win Percentage

- Sort by Win Percentage
- No consideration of number of games
- No consideration of strength of schedule
- Is "Too Drunk to Fail" really a top 5 team?
- Who would win if "AHOC" played "SnakeCountryBromance"?

Team Name	Division	Wins	Losses	Win Percentage
AHOC	Div 1	14	0	1.000000
SnakeCountryBromance	Div 1	5	0	1.000000
Injustice League	Div 2	14	1	0.933333
Pink Flamingos	Div 2	10	1	0.909091
Too Drunk to Fail	Div 3	11	2	0.846154
Jack's Abby HAOS Lager	Div 2	10	2	0.833333
Maverick	Div 2	9	2	0.818182
JuJu Hex	Div 2	11	3	0.785714
Upstream	Div 2	20	6	0.769231
Gothrilla	Div 1	12	4	0.750000

Ranking Teams: A Bayesian Framework

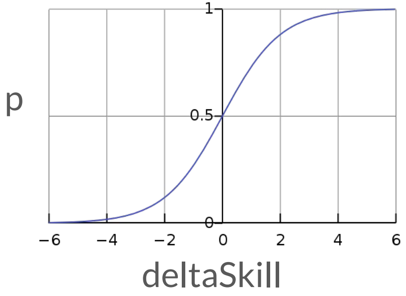
- Suppose each team has an inherent true skill level, denoted as "`skill`"
- Consider two teams, Team A and Team B, with skill levels denoted as `skillA` and `skillB`, respectively
- In our model, what matters is the *difference* in skill levels, `delta_skill`:
  `delta_skill = skillA - skillB`
- We use the logit function to convert `deltaSkill` to `p`, the probability that Team A beats Team B
  `p = 1 / (1 + exp(-delta_skill))`



A Bayesian Framework: An Example

- Example
    - `skillA = 4.0, skillB = 2.0`
    - `delta_skill = 2.0`
    - `p = 1 / (1 + exp(-delta_skill) = 1 / (1 + exp(-2)) = 88.1%`

=> We expect Team A to beat Team B 88.1% of the time



A Bayesian Framework: The Data

- An ultimate frisbee league has `N` teams and `M` game outcomes
- We will use 2016 Boston Ultimate Disc Alliance summer league data with `N=67` and `M=522`
- `skills` is a vector of length `N` containing `skill` for each team in the league
- Each game outcome is binary (i.e., ignore ties) and is denoted "`Team A Wins`"
- We relate p to the observed outcomes using the Bernoulli distribution

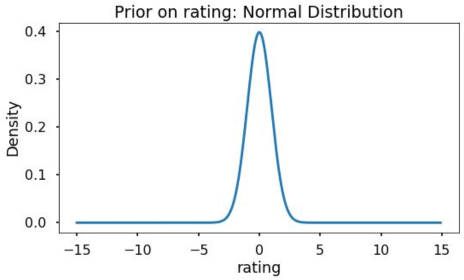
Team A	Team B	Team A Wins	Div A	Div B
AHOC	Tubbs	True	Div 1	Div 2
AHOC	Lady and the BAMF	True	Div 1	Div 2
AHOC	Live Poultry, Fresh Killed (LPFK)	True	Div 1	Div 1
AHOC	BBN	True	Div 1	Div 1
AHOC	JuJu Hex	True	Div 1	Div 2
AHOC	Turtle Boy	True	Div 1	Div 1
AHOC	Upstream	True	Div 1	Div 2
AHOC	TuneSquad	True	Div 1	Div 1
AHOC	Swingers	True	Div 1	Div 1
AHOC	Stonecutters	True	Div 1	Div 1
AHOC	Stonecutters	True	Div 1	Div 1
AHOC	FlowChart	True	Div 1	Div 1
AHOC	Gothrilla	True	Div 1	Div 1
AHOC	Zerg Rush!	True	Div 1	Div 1

Bayes Rule

$$\underbrace{P(\text{skills} \mid \text{Team A Wins})}_{\text{Posterior}} \propto \underbrace{P(\text{skills})}_{\text{Prior}} \; \underbrace{P(\text{Team A Wins} \mid \text{skills})}_{\text{Likelihood}}$$

Priors

- To start out, let’s assume we know nothing about each team’s skill before any games have been played
- Every team is “average”
  - Normal distribution
  - Mean = 0
  - Standard deviation = 1



In PyMC3:

```
with pm.model() as model:
    skills = pm.Normal(
        ‘skills’, mu=0, sd=1, shape=n_teams)

    delta_skill = skills[indexA] - skills[indexB]

    p = 1 / (1 + np.exp(-delta_skill))

    win = pm.Bernoulli(‘win’, p,
                       observed=team_A_wins)
```

- The keyword `shape` means we will get a vector of skills, one for each team in the dataframe.
- `n_teams` is the number of teams in the database

Building delta_skill

- We need `delta_skill` for every row in the dataframe (i.e., for every game in the season)
- Key insight: we can index `skills` just like a normal python list
- Plan: assign a number to each team and use that number to index `skills`

Team A	Team B	Team A Wins	Index A	Index B
AHOC	Gothrilla	True	0	1
AHOC	BBN	True	0	9
AHOC	Stonecutters	True	0	41
AHOC	FlowChart	True	0	2
AHOC	Lady and the BAMF	True	0	28

In PyMC3:

```
with pm.model() as model:
    skills = pm.Normal(
        'skills', mu=0, sd=1, shape=n_teams)

    delta_skill = skills[indexA] - skills[indexB]

    p = 1 / (1 + np.exp(-delta_skill))

    win = pm.Bernoulli('win', p,
                       observed=team_A_wins)
```

Calculating p

- p = sigmoid(delta_skill)
- sigmoid = 1 / (1 + exp(-x))

In PyMC3:

```
with pm.model() as model:
    skills = pm.Normal(
        ‘skills’, mu=0, sd=1, shape=n_teams)

    delta_skill = skills[indexA] - skills[indexB]

    p = 1 / (1 + np.exp(-delta_skill))

    win = pm.Bernoulli(‘win’, p,
                            observed=team_A_wins)
```

Likelihood Distribution

- `P(Team A Wins | skills)` means "What is the likelihood that team A won, given the skill differential between team A and B and our function that converts that differential to the probability that A wins"?
- `Team A Wins` is a binary outcome
- We use the Bernoulli distribution to model our likelihood. This distribution links a probability to a binary outcome.
  Bernoulli Distribution: http://docs.pymc.io/api/distributions/discrete.html#pymc3.distributions.discrete.Bernoulli

In PyMC3:

```
with pm.model() as model:
    skills = pm.Normal(
        'skills', mu=0, sd=1, shape=n_teams)

    delta_skill = skills[indexA] - skills[indexB]

    p = 1 / (1 + np.exp(-delta_skill))

    win = pm.Bernoulli('win', p,
                       observed=team_A_wins)
```

That's it! We've specified our model. Now we can sample and get the posterior. Let's go to a notebook to see what happens!

Let’s test it!

First, we’ll look at a Jupyter notebook for an example of a simulated season with only 3 teams of known rating.

Conclusion

- We looked at different approaches to ranking teams in BUDA ultimate frisbee leagues
- Heuristic approaches are kind of sort of ok, but suffer from problems:
  - subjective
  - offer no clear path to validation
  - can't be extended in interesting ways
- In contrast, our Bayesian approach is quantitative, offers direct validation, and can be extended in really interesting ways (e.g., what is the likelihood of AHOC beating SnakeCountryBromance?)
- We looked at two different priors and showed that they have a significant impact on the results
  - This is ok because we stated at the outset very clearly what our priors were

Next steps

- Ideas
    - Simulate the end-of-season tournament
    - Use accuracy, logloss, or some other metric to optimize the model (watch out for overfitting!)
    - Use data from previous years:
        - to specify prior for this year
        - to implement cross-validation
    - Incorporate goals-for and goals-against into analysis (Binomial distribution)
- Game scores (for all seasons through 2016) and jupyter notebooks are available
- I’m very interested to see what you come up with!
    - sbussmann@cibotechnologies.com
- CiBO is hosting a Stan meetup on July 24. Please sign up and attend if you have an interest in learning Stan (note: Stan is a probabilistic programming language similar to PyMC3)!

Resources

- http://camdavidsonpilon.github.io/Probabilistic-Programming-and-Bayesian-Methods-for-Hackers/
- PyMC3: http://docs.pymc.io/#learn-bayesian-statistics-with-a-book-together-with-pymc3
- PyMC3 discussion forum (very active, helpful userbase): https://discourse.pymc.io/