Ranking Ultimate Frisbee Teams

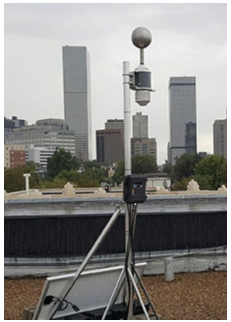
Insight Bayesian Workshop

Shane Bussmann
February 20, 2018

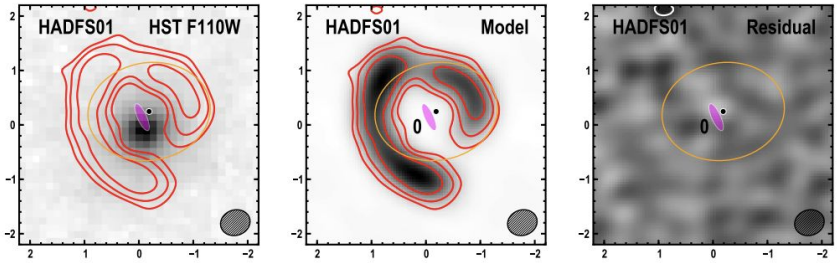
About Me

- 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

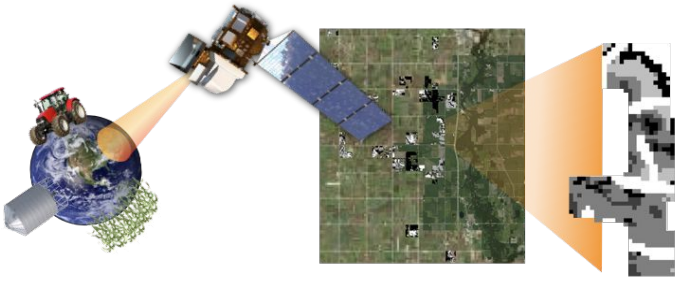


CiBO Technologies is

Planetary-Scale

Agricultural

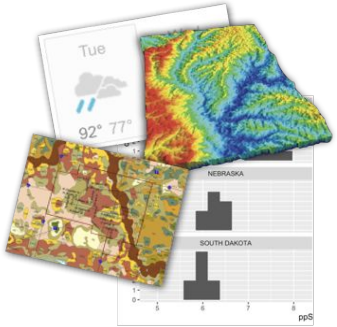
Simulation Optimization



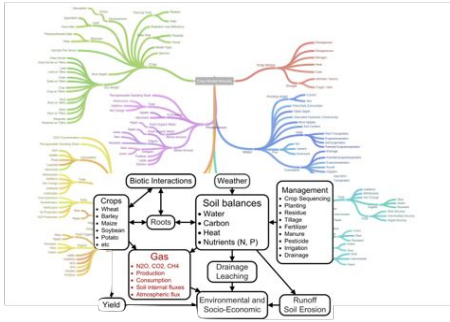
Planetary Scale, Daily Simulations

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
- Teams self-assign to one of four 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)
- Some teams play with 4 men and 3 women, others with 5 men and 2 women
- We have been asked to rank teams for the end-of-season tournament

Ranking Teams: Plus/Minus?

- Sort by Plus/Minus
- No consideration of strength of schedule
- Is Upstream really better than AHOC?
- Who would win if they played each other?

Team Name	PlusMinus	divname	Wins	Losses	Ties	winpercent
Upstream	119	4/3 Div 2	20	6	0	0.769231
AHOC	100	4/3 Div 1	14	0	0	1.000000
Injustice League	100	4/3 Div 2	14	1	0	0.933333
Puddingstone	98	4/3 Div 2	18	6	0	0.750000
Crossroads	73	Open Div 1	17	3	1	0.850000
JuJu Hex	72	5/2 Div 2	11	3	1	0.785714
Flaming Croissants	66	4/3 Div 2	15	5	0	0.750000
Too Drunk to Fail	59	4/3 Div 3	11	2	0	0.846154
Pink Flamingos	58	5/2 Div 2	10	1	0	0.909091
Baba Yaga	47	5/2 Div 2	9	3	0	0.750000
Gothrilla	47	4/3 Div 1	12	4	1	0.750000
Jack's Abby HAOS Lager	46	4/3 Div 2	10	2	0	0.833333
License to Kilt (fka Scoobers in Scotland)	46	4/3 Div 2	13	6	0	0.684211

Ranking Teams: Division + Plus/Minus

- Sort by division
- Then by Plus/Minus

→ Worst Div 1 team > best Div 2 team

- Still ignoring strength of schedule
- Are Upstream and Injustice League really worse than BBN and LPFK?

Team Name	PlusMinus	divname	Wins	Losses	Ties	winpercent
AHOC	100	4/3 Div 1	14	0	0	1.000000
Gothrilla	47	4/3 Div 1	12	4	1	0.750000
FlowChart	29	4/3 Div 1	12	4	0	0.750000
GrassBurner	29	4/3 Div 1	11	8	2	0.578947
Zerg Rush!	22	4/3 Div 1	10	4	0	0.714286
Swingers	-6	4/3 Div 1	5	7	1	0.416667
TuneSquad	-9	4/3 Div 1	6	7	1	0.461538
Turtle Boy	-12	4/3 Div 1	5	10	0	0.333333
Live Poultry, Fresh Killed (LPFK)	-39	4/3 Div 1	6	13	0	0.315789
BBN	-47	4/3 Div 1	2	8	1	0.200000
Upstream	119	4/3 Div 2	20	6	0	0.769231
Injustice League	100	4/3 Div 2	14	1	0	0.933333
Puddingstone	98	4/3 Div 2	18	6	0	0.750000
Flaming Croissants	66	4/3 Div 2	15	5	0	0.750000
Jack's Abby HAOS Lager	46	4/3 Div 2	10	2	0	0.833333
License to Kilt (fka Scoobers in Scotland)	46	4/3 Div 2	13	6	0	0.684211

Ranking Teams: A Bayesian Framework

- Suppose each team has a “rating” that represents the true skill level of that team
- Consider two teams, Team A and Team B, with ratings denoted as `ratingA` and `ratingB`, respectively
  - The difference in ratings is `deltaRating = ratingA - ratingB`
  - The probability that Team A wins is `p`
  - `p` is a function of `deltaRating`: `p = f(deltaRating)`
- In our data, let’s assume that the outcome is binary (i.e., ignore ties) and is denoted “`Team A Wins`”
- Bayes rule tells us how to use our measurements (`Team A Wins`) to update our belief about `deltaRating` given our model for `p`

Team A	Team B	divname	Team A Wins
AHOC	Gothrilla	4/3 Div 1	True
AHOC	BBN	4/3 Div 1	True
AHOC	Stonecutters	4/3 Div 1	True
AHOC	FlowChart	4/3 Div 1	True
AHOC	Lady and the BAMF	4/3 Div 1	True
AHOC	Swingers	4/3 Div 1	True
AHOC	Upstream	4/3 Div 1	True
AHOC	Tubbs	4/3 Div 1	True
AHOC	Stonecutters	4/3 Div 1	True
AHOC	Zerg Rush!	4/3 Div 1	True
AHOC	Turtle Boy	4/3 Div 1	True
AHOC	Live Poultry, Fresh Killed (LPFK)	4/3 Div 1	True
AHOC	TuneSquad	4/3 Div 1	True
AHOC	JuJu Hex	4/3 Div 1	True
Gothrilla	AHOC	4/3 Div 1	False
Gothrilla	Swingers	4/3 Div 1	False

Ranking Teams: A Bayesian Framework

- Posterior (what we want)
  `P(f(deltaRating) | Team A Wins)`
- Prior (where we start)
  `P(f(deltaRating))`
- Likelihood (how we go from the prior to the posterior)
  `P(Team A Wins | f(deltaRating))`
  “What is the likelihood that team A won, given the rating differential between team A and B and our function that converts that differential to the probability that A wins”
- Support:
  `Likelihood / P(Team A Wins) = Likelihood / 0.5`

Team A	Team B	divname	Team A Wins
AHOC	Gothrilla	4/3 Div 1	True
AHOC	BBN	4/3 Div 1	True
AHOC	Stonecutters	4/3 Div 1	True
AHOC	FlowChart	4/3 Div 1	True
AHOC	Lady and the BAMF	4/3 Div 1	True
AHOC	Swingers	4/3 Div 1	True
AHOC	Upstream	4/3 Div 1	True
AHOC	Tubbs	4/3 Div 1	True
AHOC	Stonecutters	4/3 Div 1	True
AHOC	Zerg Rush!	4/3 Div 1	True
AHOC	Turtle Boy	4/3 Div 1	True
AHOC	Live Poultry, Fresh Killed (LPFK)	4/3 Div 1	True
AHOC	TuneSquad	4/3 Div 1	True
AHOC	JuJu Hex	4/3 Div 1	True
Gothrilla	AHOC	4/3 Div 1	False
Gothrilla	Swingers	4/3 Div 1	False

Ranking Teams: A Bayesian Framework

- Posterior (what we want)
  `P(f(deltaRating) | Team A Wins)`
- Prior (where we start)
  `P(f(deltaRating))`
- Likelihood (how we go from the prior to the posterior)
  `P(Team A Wins | f(deltaRating))`
  “What is the likelihood that team A won, given the rating differential between team A and B and our function that converts that differential to the probability that A wins”
- Support:
  `Likelihood / P(Team A Wins) = Likelihood / 0.5 ~ Likelihood`

Team A	Team B	divname	Team A Wins
AHOC	Gothrilla	4/3 Div 1	True
AHOC	BBN	4/3 Div 1	True
AHOC	Stonecutters	4/3 Div 1	True
AHOC	FlowChart	4/3 Div 1	True
AHOC	Lady and the BAMF	4/3 Div 1	True
AHOC	Swingers	4/3 Div 1	True
AHOC	Upstream	4/3 Div 1	True
AHOC	Tubbs	4/3 Div 1	True
AHOC	Stonecutters	4/3 Div 1	True
AHOC	Zerg Rush!	4/3 Div 1	True
AHOC	Turtle Boy	4/3 Div 1	True
AHOC	Live Poultry, Fresh Killed (LPFK)	4/3 Div 1	True
AHOC	TuneSquad	4/3 Div 1	True
AHOC	JuJu Hex	4/3 Div 1	True
Gothrilla	AHOC	4/3 Div 1	False
Gothrilla	Swingers	4/3 Div 1	False

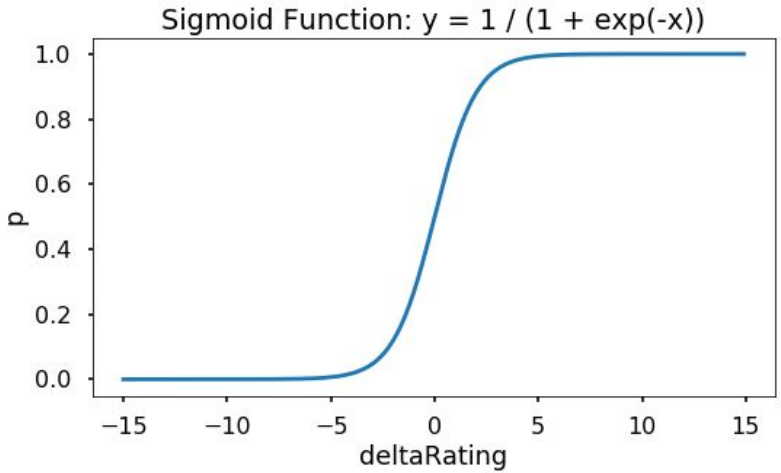
`P(f(deltaRating) | Team A Wins) = P(f(deltaRating)) * P(Team A Wins | f(deltaRating))`

p = f(deltaRating)

- First, we need to define the function that converts rating differential between two teams into a probability that the first team wins
- Key requirements
  - If deltaRating << 0, p = 0 (i.e., Team B wins always)
  - If deltaRating >> 0, p = 1 (i.e., Team A wins always)
  - If deltaRating == 0, p = 0.5
- Ideas?

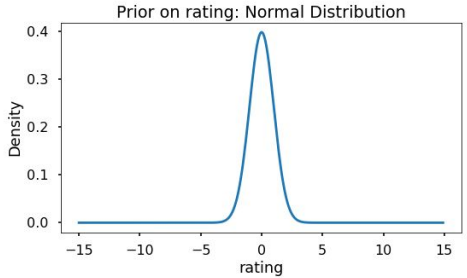
p = f(deltaRating)

- First, we need to define the function that converts rating differential between two teams into a probability that the first team wins
- Key requirements
  - If deltaRating << 0, p = 0 (i.e., Team B wins always)
  - If deltaRating >> 0, p = 1 (i.e., Team A wins always)
  - If deltaRating == 0, p = 0.5
- sigmoid function
  - y = 1 / (1 + exp(-x))
  - jupyter notebook: Insight Bayesian Workshop - Sigmoid



Priors

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



In PyMC3:

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

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

Building deltaRating

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

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:
    ratings = pm.Normal(
        'ratings', mu=0, sd=1, shape=nTeams)

    deltaRating = ratings[indexA] -
                      ratings[indexB]
```

Implementing sigmoid

- p = sigmoid(deltaRating)
- Recall that sigmoid = 1 / (1 + exp(-x))

In PyMC3:

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

    deltaRating = ratings[indexA] -
                      ratings[indexB]

    p = 1 / (1 + np.exp(-deltaRating))
```

Likelihood Distribution

- `P(Team A Wins | f(deltaRating))` means “What is the likelihood that team A won, given the rating 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
- What distribution should we use to model our likelihood? What distribution links a probability to a binary outcome?

In PyMC3:

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

    deltaRating = ratings[indexA] -
                      ratings[indexB]

    p = 1 / (1 + np.exp(-deltaRating))
```

Likelihood Distribution

- `P(Team A Wins | f(deltaRating)` means “What is the likelihood that team A won, given the rating 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
- What distribution should we use to model our likelihood? What 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:
    ratings = pm.Normal(
        ‘ratings’, mu=0, sd=1, shape=nTeams)

    deltaRating = ratings[indexA] -
                      ratings[indexB]

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

    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!

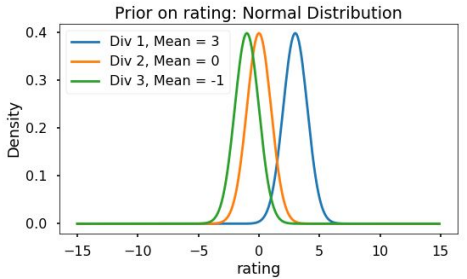
But first, we’ll look at a Jupyter notebook for an example of a simulated season with only 3 teams of known rating.

What went wrong?

- Got the top 2 teams right, but many other instances of questionable results
- Ideas for changes?

What went wrong?

- Got the top 2 teams right, but many other instances of questionable results
- Ideas for changes?
  - Support?
    - Bernoulli distribution is definitely the right way to convert probability to binary outcome
    - Sigmoid may not be the best way to convert rating differential to probability, but it seems reasonable
  - Priors?
    - We expect that a Div 1 team should beat a Div 2 team. Maybe we should try to incorporate that prior knowledge into our model?
    - One possible approach:
      - Avg. Div 1 beats Avg. Div 2 about 95% of the time
      - Avg. Div 2 beats Avg. Div 3 about 75% of the time



Let’s test it!

Switch to Jupyter notebook.

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 team A winning the tournament?)
- 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 to you
- I'm very interested to see what you come up with!
  - sbussmann@cibotechnologies.com
- I will be at Boston Bayesians meetup on Feb 22, hope to see you there!

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/

p = f(deltaRating, sharpness)

- Next, we need to add a parameter that characterizes how sharp the transition is from p=0 to p=0.5 and from p=0.5 to p=1.0
- We will call this "`sharpness`"
- Higher sharpness => faster transition from p=0 to p=0.5 and from p=0.5 to p=1.0
- Lower sharpness => slower transition from p=0 to p=0.5 and from p=0.5 to p=1.0
- sigmoid function with sharpness
  - y = (1 + exp(-sharpness * x)
  - jupyter notebook: Insight Bayesian Workshop - Sigmoid