Fully Bayesian Models

Bayesian Idea: Treat a parameter as having a distribution to be estimated, rather than as an unknown fixed number to be estimated

Ex 1 Predict 2nd half-of-season win percentage from mid-season wins and losses

$$\text{Beta-Binomial} \begin{cases} W \sim \text{Binomial}(m, p) \\ p \sim \text{Beta}(\alpha, \beta) \end{cases}$$

When we modeled the team's latent game win probability p by a Beta distribution, we encode the prior information that p is more likely to be near $\frac{\alpha}{\alpha+\beta}$ ($\alpha = 15, \beta = 15$, $\frac{1}{2}$) than near the extremes 0 or 1.

Then we found by Bayes Rule that the posterior distribution $p \mid W, L$ is

$$p \mid W, L \sim \text{Beta}(\alpha + W, \quad \beta + L) \;\; ?$$

and the Bayes estimate of p is the posterior mean

$$\hat{p}^{(\text{Bayes})} = \mathbb{E}[p \mid W, L] = \frac{\alpha + W - 1}{W + L + \alpha + \beta - 2}$$

Ex 2 Predict 2nd-half-of-season Batting Average from Mid-season batting average and number of at bats

Normal-Normal Model

$$\begin{cases} i = \text{player } i \\ H_i = \#\text{hits}, \; N_i = \#\text{ at bats}, \; X_i = \frac{H_i}{N_i} \\ X_i \sim N(p_i, \sigma_i^2) \rightarrow \text{CLT} \\ \sigma_i^2 = c/N_i \quad \text{Known} \\ p_i \sim N(\mu, \tau^2) \end{cases}$$

When we modeled player i's latent batting quality p_i using a Normal distribution, we encoded the prior information that player i is a baseball player who is more likely to be close to μ than to some extreme number like 0 or 1.

Then we could use Bayes Rule to solve for the posterior distribution

$$P_i \mid X_i, N_i \sim \mathcal{N}\left(\frac{\frac{X_i}{C/N_i} + \frac{\mu}{\tau^2}}{\frac{1}{C/N_i} + \frac{1}{\tau^2}}, \frac{1}{\frac{1}{C/N_i} + \frac{1}{\tau^2}}\right)$$

and the Bayes estimate is the posterior mean

$$\hat{P}_i^{(Bayes)} = \mathbb{E}(P_i \mid X_i, N_i) = \frac{\frac{X_i}{C/N_i} + \frac{\hat{\mu}}{\hat{\tau}^2}}{\frac{1}{C/N_i} + \frac{1}{\hat{\tau}^2}}$$

which is expressed in terms of unknown hyperparameters μ, τ^2 so we introduced Empirical Bayes

Ex 3 Bayesian Regression

Model
- Regression $y_i \sim \mathcal{N}(\vec{X}_i \cdot \vec{\beta}, \sigma^2)$
- Prior $\vec{\beta} \sim \mathcal{N}(\vec{0}, \frac{\sigma^2}{\lambda} \cdot I)$

$\beta_j \overset{ind}{\sim} N(0, \frac{\sigma^2}{\lambda})$

If you use Bayes Rule to find the posterior dist $P(\beta | X, y)$ you'll find

$$\beta \sim N\left((X^T X + \lambda \cdot I)^{-1} \cdot X^T y, \quad \_\_ \right)$$

Multivariable Regression: $\hat{\beta} = (X^T X)^{-1} X^T y$

This version of Bayesian Regression: $\hat{\beta} = (X^T X + \lambda \cdot I)^{-1} X^T y$

Ridge Regression

pretend X is a number and not a matrix

$$\frac{1}{x^2 + \lambda} \cdot (xy)$$

versus

$$\frac{1}{x^2} \cdot xy$$

$X^T X = U D U^T$

D diagonal all positive entries

$X^T X + \lambda I = U (D + \lambda) U^T$

Bayesian Modeling: your goal is to estimate a full posterior distribution on the parameters of interest.

↳ Why estimate the full posterior dist and not just the Bayes estimate (posterior mean) $\hat{\beta}$ or $\hat{p}$?

You want some measure of how strongly you believe in your estimate.
You want to measure uncertainty in your estimate.

And in the 3 previous examples, we were able to find closed-form solutions for the posterior, because we used simple models with easy/nice prior distributions.
Often it is impossible to find a formula for the posterior :(

Create a fully Bayesian NFL Power Rating Model

Dataframe:

game_id	home_team	away_team	season_type	week	total_home_score	total_away_score	season	pts_H_minus_A
2018_01_ATL_PHI	PHI	ATL	REG	1	18	12	2018	6
2018_01_BUF_BAL	BAL	BUF	REG	1	47	3	2018	44
2018_01_CHI_GB	GB	CHI	REG	1	24	23	2018	1
2018_01_CIN_IND	IND	CIN	REG	1	23	34	2018	-11
2018_01_DAL_CAR	CAR	DAL	REG	1	16	8	2018	8
2018_01_HOU_NE	NE	HOU	REG	1	27	20	2018	7
2018_01_JAX_NYG	NYG	JAX	REG	1	15	20	2018	-5
2018_01_KC_LAC	LAC	KC	REG	1	28	38	2018	-10
2018_01_LA_OAK	LV	LA	REG	1	13	33	2018	-20
2018_01_NYJ_DET	DET	NYJ	REG	1	17	48	2018	-31
2018_01_PIT_CLE	CLE	PIT	REG	1	21	21	2018	0
2018_01_SEA_DEN	DEN	SEA	REG	1	27	24	2018	3

```
# A tibble: 1,657 × 13
   game_id         home_team away_team season_type  week total_home_score total_away_score season pts_H_minus_A     S     H     A     y
   <chr>           <chr>     <chr>     <chr>       <dbl>            <dbl>            <dbl>  <dbl>         <dbl> <dbl> <int> <int> <dbl>
 1 2018_01_ATL_PHI PHI       ATL       REG             1               18               12   2018             6     1    26     2     6
 2 2018_01_BUF_BAL BAL       BUF       REG             1               47                3   2018            44     1     3     4    44
 3 2018_01_CHI_GB  GB        CHI       REG             1               24               23   2018             1     1    12     6     1
 4 2018_01_CIN_IND IND       CIN       REG             1               23               34   2018           -11     1    14     7   -11
 5 2018_01_DAL_CAR CAR       DAL       REG             1               16                8   2018             8     1     5     9     8
 6 2018_01_HOU_NE  NE        HOU       REG             1               27               20   2018             7     1    22    13     7
 7 2018_01_JAX_NYG NYG       JAX       REG             1               15               20   2018            -5     1    24    15    -5
 8 2018_01_KC_LAC  LAC       KC        REG             1               28               38   2018           -10     1    18    16   -10
 9 2018_01_LA_OAK  LV        LA        REG             1               13               33   2018           -20     1    19    17   -20
10 2018_01_NYJ_DET DET       NYJ       REG             1               17               48   2018           -31     1    11    25   -31
# ℹ 1,647 more rows
```

Variables

Row i = game i

$H(i)$ = index of the home team in game i

$A(i)$ = index away

$S(i)$ = index season

y_i = pts scored by $H(i)$ − pts scored by $A(i)$

Model

OLD: $y_i = \beta_0 + \beta_{H(i),S(i)} - \beta_{A(i),S(i)} + \varepsilon_i$

Bayesian:

$$\begin{cases} y_i \sim \mathcal{N}\left(\beta_0 + \beta_{H(i),S(i)} - \beta_{A(i),S(i)},\ \sigma^2_{game}\right) & \\ \beta_{j,s} \sim \mathcal{N}\left(\gamma \cdot \beta_{j,s-1},\ \sigma^2_{season}\right) & \forall j\ \forall s>1 \\ \beta_{j,1} \sim \mathcal{N}\left(0,\ \sigma^2_{teams}\right) & \forall j \\ \beta_0 \sim \mathcal{N}(0, 5^2) & \\ \sigma^2_{game} \sim \mathcal{N}_+(0, 5^2) & \\ \sigma^2_{season} \sim \mathcal{N}_+(0, 5^2) & \\ \sigma^2_{team} \sim \mathcal{N}_+(0, 5^2) & \\ \gamma \sim \text{Unif}[0,1] & \end{cases}$$

In a general Bayesian model like this one, the model can be large, complicated, the priors and likelihoods may not fit together nicely (conjugate), and we usually cannot do the Bayes Rule on paper to get a posterior distribution as a closed-form solution.

↓

We need to approximate the posterior distribution using MCMC (markov chain monte carlo) methods like

- Gibbs Sampling
- Hamiltonian Monte Carlo
- NUTS (no U Turn Sampling)

take Shane/Paul Bayesian class

↓

use the coding language Stan to approximate the posterior distribution.

Using Stan

- write the full Bayesian model in Stan code
- format the dataset as a dataframe in R or Python to match the Stan code
- Call the Stan sampler from RStan or PyStan which runs the Stan code to approximate the posterior distribution
- Returns posterior dist. for each parameter via posterior samples

ex posterior of β_j is approximated by samples $\{\beta_j^{(1)}, \ldots, \beta_j^{(N)}\}$ (histogram)

Fit a fully Bayesian model using Stan

"bayesian_model_glickmanStern.stan"

```
data {
  int<lower=1> N_games;            // number of games
  int<lower=1> N_teams;            // number of teams
  int<lower=2> N_seasons;          // number of seasons

  real y[N_games];                 // outcome vector (point differential)
  int<lower=1, upper=N_teams> H[N_games];   // vector of home team indices
  int<lower=1, upper=N_teams> A[N_games];   // vector of away team indices
  int<lower=1, upper=N_seasons> S[N_games]; // vector of season indices
}
```

```
parameters {
  real beta_0;                        // intercept (home field advantage)
  real betas[N_teams, N_seasons];   // team strength coefficients for each team-season

  real<lower=0> sigma_games;       // game-level variance in point differential
  real<lower=0> sigma_teams;       // variance across teams before the first season
  real<lower=0> sigma_seasons;     // a team's variance across seeasons
  real<lower=0, upper=1> gamma;    // autoregressive parameter
}
```

```
model {
  // game-level model
  for (i in 1:N_games) {
    y[i] ~ normal(beta_0 + betas[H[i],S[i]] - betas[A[i],S[i]], sigma_games);
  }

  // team-level priors
  for (j in 1:N_teams) {
    // initial season prior across teams
    betas[j,1] ~ normal(0, sigma_teams);
    for (s in 2:N_seasons) {
      // auto-regressive model across seasons
      betas[j,s] ~ normal(gamma*betas[j,s-1], sigma_seasons);
    }
  }

  // priors
  sigma_games ~ normal(0, 5);
  sigma_teams ~ normal(0, 5);
  sigma_seasons ~ normal(0, 5);
  gamma ~ uniform(0, 1);
}
```

Data

Row i = game i

$H(i)$ = index of the home team in game i

$A(i)$ = index away

$S(i)$ = index season

y_i = pts scored by $H(i)$ − pts scored by $A(i)$

$$
\begin{cases}
y_i \sim \mathcal{N}\left(\beta_0 + \beta_{H(i),S(i)} - \beta_{A(i),S(i)},\ \sigma^2_{game}\right) & \forall i \\
\beta_{j,s} \sim \mathcal{N}\left(\gamma \cdot \beta_{j,s-1},\ \sigma^2_{season}\right) & \forall j,\ \forall s>1 \\
\beta_{j,1} \sim \mathcal{N}\left(0,\ \sigma^2_{teams}\right) & \forall j \\
\beta_0 \sim \mathcal{N}(0, 5^2) \\
\sigma^2_{game} \sim \mathcal{N}_+(0, 5^2) \\
\sigma^2_{season} \sim \mathcal{N}_+(0, 5^2) \\
\sigma^2_{team} \sim \mathcal{N}_+(0, 5^2) \\
\gamma \sim \text{Unif}[0,1]
\end{cases}
$$

Go into R

library(rstan)

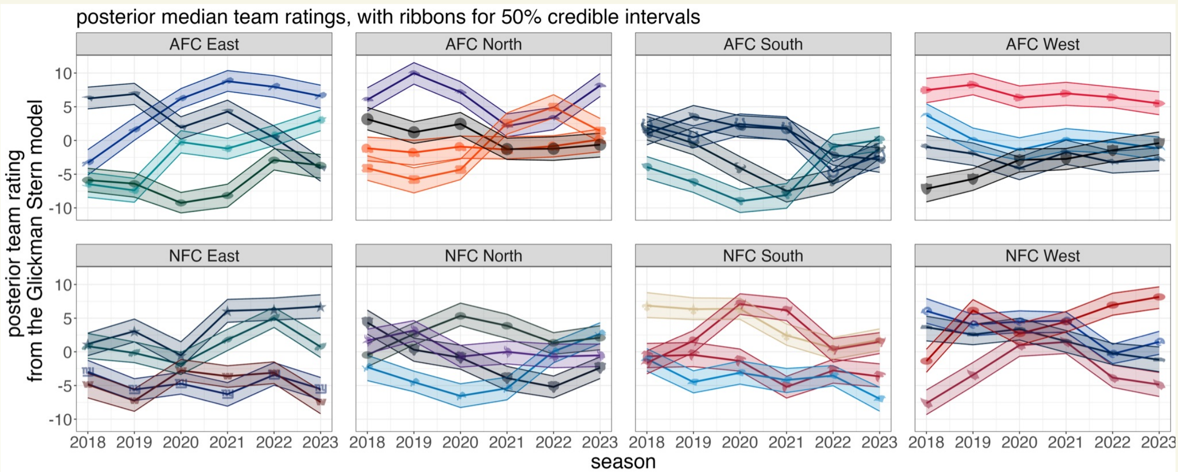
```
### load stan model
MODEL <- stan_model(file = "bayesian_model_glickmanStern.stan", model_name = "glickmanSternModel")
MODEL

### create list of data compliant with the Stan model
data_train <- list(
  N_games = nrow(df1),
  N_teams = nrow(map_team_to_idx),
  N_seasons = length(unique(df1$season)),
  y = df1$y,
  H = df1$H,
  A = df1$A,
  S = df1$S
)
data_train

# Train the model
fit <- sampling(
  MODEL, data = data_train, iter = 1500, chains = 1, seed = 12345,
)
fit
```

Results

	param	post_lower	post_med	post_upper
	<chr>	<dbl>	<dbl>	<dbl>
1	beta_0	0.968	1.53	2.08
2	sigma_games	12.0	12.4	12.9
3	sigma_teams	3.70	4.94	6.75
4	sigma_seasons	3.02	3.70	4.49
5	gamma	0.493	0.662	0.814



There are <u>many</u> great sports papers that use fully Bayesian models because they

- are interpretable
- quantify uncertainty
- capture multiple sources of variation
- use shrinkage via prior