Mini Project 3

Brenden Bready

2024-04-08

```
# Loading in data
twentytwo = load_pbp(2022)
twentythree = load_pbp(2023)
data = rbind(twentytwo, twentythree)
```

1. EPA per Play

For EPA per play, I first filtered out the plays in which there was no team as labeled the possession team (beginning of a quarter, end of a quarter, etc). After this, I grouped by the game id, teams, and whether or not it was the special teams on the field to then calculate the EPA per play and the total EPA. Here, the total EPA is just the EPA per play * Number of plays. A sample of this dataset, which will be used later to calculate the power rankings, can be seen below.

```
# Create EPA dataset
epa_data = data %>%
  filter(!is.na(posteam)) %>%
  group_by(game_id, season, home_team, away_team,
           posteam, defteam, special) %>%
  mutate_at(vars(epa), ~ replace_na(., 0)) %>% # Change NA to 0
  summarise(epa_total = sum(epa),
            epa_play = mean(epa),
            plays = n()) %>%
  ungroup()
```

Table 1: First rows of EPA per Play Dataset

game_id	home_team	away_team	posteam	defteam	special	epa_total	epa_play	plays
2022_01_BAL_NYJ	NYJ	BAL	BAL	NYJ	0	0.22	0.00	60
2022_01_BAL_NYJ	NYJ	BAL	BAL	NYJ	1	-9.86	-0.66	15
2022_01_BAL_NYJ	NYJ	BAL	NYJ	BAL	0	-14.76	-0.17	89
2022_01_BAL_NYJ	NYJ	BAL	NYJ	BAL	1	-10.33	-0.86	12
2022_01_BUF_LA	LA	BUF	BUF	LA	0	8.51	0.13	64
2022_01_BUF_LA	LA	BUF	BUF	LA	1	-14.04	-1.28	11

2. Bradley-Terry Rankings

To start, we need to set up the dataset to get the final results from each game to calculate wins and losses. This can be seen below.

```r
finalScores = data %>%
  group_by(game_id) %>%
  select(game_id, season, home_team, away_team, total_home_score, total_away_score) %>%
  slice(n())
kable(head(finalScores), caption = "First rows of Final Scores dataset")
```

Table 2: First rows of Final Scores dataset

game_id	season	home_team	away_team	total_home_score	total_away_score
2022_01_BAL_NYJ	2022	NYJ	BAL	9	24
2022_01_BUF_LA	2022	LA	BUF	10	31
2022_01_CLE_CAR	2022	CAR	CLE	24	26
2022_01_DEN_SEA	2022	SEA	DEN	17	16
2022_01_GB_MIN	2022	MIN	GB	23	7
2022_01_IND_HOU	2022	HOU	IND	20	20

Now, I can add a column to indicate whether the home team won, away team won, or it was a tie. From here, I can manipulate the dataset to get the wins of each team against won another home and away.

```r
# First, add column to indicate home win, away win, or tie
finalScores = finalScores %>%
  mutate(home_win = ifelse(total_home_score > total_away_score, 1, 0),
         away_win = ifelse(total_away_score > total_home_score, 1, 0),
         tie = ifelse(total_home_score == total_away_score, 1, 0))

# Change dataset to get number of wins of each team against one another
wins = finalScores %>%
  group_by(home_team, away_team, season) %>%
  summarise(home_wins = sum(home_win), away_wins = sum(away_win), ties = sum(tie)) %>%
  ungroup(home_team, away_team)
kable(head(wins), caption = "First rows of Wins dataset")
```

Table 3: First rows of Wins dataset

home_team	away_team	season	home_wins	away_wins	ties
ARI	ATL	2023	1	0	0
ARI	BAL	2023	0	1	0
ARI	CIN	2023	0	1	0
ARI	DAL	2023	1	0	0
ARI	KC	2022	0	1	0
ARI	LA	2022	0	1	0

To run the Bradley Terry model, I first have to change my teams to factors. Since I need the model in terms of wins and losses, I will add all ties as half a win for each team. Below I fit the model for both the 2022 and 2023 seasons without home field advantage, and a summary of the output for the 2022 season is shown below.

```r
# Changing to factors so model runs properly
wins$home_team = as.factor(wins$home_team)
wins$away_team = as.factor(wins$away_team)
```

```r
# Add ties as half a win
wins = wins %>%
  mutate(home_wins = home_wins + 0.5*ties,
         away_wins = away_wins + 0.5*ties)

# Separate seasons
wins2022 = wins %>%
  filter(season == 2022)
wins2023 = wins %>%
  filter(season == 2023)

# Fit Bradley Terry for 2022
bt22mod = BTm(cbind(home_wins, away_wins), home_team, away_team,
                  data = wins2022, id = "team")

# Fit Bradley Terry for 2023
bt23mod = BTm(cbind(home_wins, away_wins), home_team, away_team,
                  data = wins2023, id = "team")

summary(bt22mod)
```

```
##
## Call:
## BTm(outcome = cbind(home_wins, away_wins), player1 = home_team,
##     player2 = away_team, id = "team", data = wins2022)
##
## Coefficients:
##         Estimate Std. Error z value Pr(>|z|)
## teamATL   0.8394     0.7874   1.066 0.286442
## teamBAL   1.8940     0.8340   2.271 0.023154 *
## teamBUF   3.0454     0.9044   3.367 0.000759 ***
## teamCAR   0.8500     0.7891   1.077 0.281380
## teamCHI   0.5294     0.9136   0.579 0.562268
## teamCIN   2.9104     0.8721   3.337 0.000846 ***
## teamCLE   1.1886     0.8389   1.417 0.156524
## teamDAL   2.5704     0.8589   2.993 0.002764 **
## teamDEN   0.1200     0.8184   0.147 0.883384
## teamDET   1.6411     0.8500   1.931 0.053515 .
## teamGB    1.5886     0.8539   1.860 0.062833 .
## teamHOU  -0.5876     0.9757  -0.602 0.547065
## teamIND   0.5193     0.8873   0.585 0.558373
## teamJAX   1.4977     0.8453   1.772 0.076428 .
## teamKC    3.3911     0.9397   3.609 0.000308 ***
## teamLA    0.2963     0.7724   0.384 0.701296
## teamLAC   1.2979     0.8050   1.612 0.106875
## teamLV    0.4006     0.8073   0.496 0.619795
## teamMIA   1.7979     0.8464   2.124 0.033662 *
## teamMIN   2.6654     0.8761   3.042 0.002348 **
## teamNE    1.4704     0.8383   1.754 0.079447 .
## teamNO    0.9454     0.7955   1.188 0.234658
## teamNYG   2.0544     0.8479   2.423 0.015392 *
## teamNYJ   1.3315     0.8481   1.570 0.116426
## teamPHI   3.2328     0.8937   3.617 0.000298 ***
```

```
## teamPIT   1.7349     0.8426   2.059 0.039480 *
## teamSEA   1.1883     0.7616   1.560 0.118689
## teamSF    2.4324     0.8244   2.950 0.003174 **
## teamTB    1.2119     0.7929   1.528 0.126415
## teamTEN   1.0105     0.8811   1.147 0.251417
## teamWAS   1.7307     0.8615   2.009 0.044543 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 388.16  on 278  degrees of freedom
## Residual deviance: 307.17  on 247  degrees of freedom
## AIC: 372.55
##
## Number of Fisher Scoring iterations: 4
```

Now, I will add to this model to account for home field advantage. This can be seen below, along with the AIC and BIC for both seasons and both models, showing that adding home field advantage helps improve our model fit.

```
# Add in information about home field advantage
wins2022$home_team <- data.frame(team = wins2022$home_team, at_home = 1)
wins2022$away_team <- data.frame(team = wins2022$away_team, at_home = -1)
wins2023$home_team <- data.frame(team = wins2023$home_team, at_home = 1)
wins2023$away_team <- data.frame(team = wins2023$away_team, at_home = -1)

# Fit the new Bradley Terry models
bt22mod2 <- update(bt22mod, formula = ~ team + at_home)
bt23mod2 <- update(bt23mod, formula = ~ team + at_home)
```

Table 4: AIC and BIC by Model and Season

Season	Advantage	AICs	BICs
2022	None	372.55	485.00
2022	Home Field	368.96	485.04
2023	None	392.69	505.48
2023	Home Field	390.25	506.68

In both seasons, we see that the AIC and BIC are both lower when we include information about the home field advantage. Now, let's look at the top 10 ranked teams from each season.

Table 5: Top 10 Teams in 2022-2023 season by Bradley Terry

team	BTRating
KC	29.70
PHI	25.35
BUF	21.02
CIN	18.36
MIN	14.37
DAL	13.07

team	BTRating
SF	11.39
NYG	7.80
BAL	6.65
MIA	6.04

Table 6: Top 10 Teams in 2023-2024 season by Bradley Terry

team	BTRating
BAL	12.21
SF	8.62
KC	7.95
DET	6.61
CLE	6.29
CIN	5.17
PIT	5.08
BUF	4.63
DAL	4.58
LA	4.50

Here, the most surprising result I immediately noticed was that the Chiefs were ranked third this year despite winning the Superbowl. However, Baltimore and San Francisco did have a better regular season record than the Chiefs, so this is an unsurprising result for a model that only considers wins and losses.

3. ELO

For the ELO ratings, I can use the final scores dataset I created as I began the portion of the Bradley Terry analysis. A sample of the data I will use can be seen below.

Table 7: Sample of Final Scores Data

game_id	home_team	away_team	home_score	away_score	home_win	away_win	tie
2022_01_BAL_NYJ	NYJ	BAL	9	24	0	1	0
2022_01_BUF_LA	LA	BUF	10	31	0	1	0
2022_01_CLE_CAR	CAR	CLE	24	26	0	1	0
2022_01_DEN_SEA	SEA	DEN	17	16	1	0	0
2022_01_GB_MIN	MIN	GB	23	7	1	0	0
2022_01_IND_HOU	HOU	IND	20	20	0	0	1

The question for the ELO ratings is what value to use for our learning rate K. To choose an appropriate, I will run the model for all values of k from 0.1 to 50 (by 0.1), and then select the model with the smallest MSE.

```
# Find the value of k with the smallest MSE
mses = c()
for(i in seq(1, 100, by = 0.2)){
  elo22mod = elo.run(score(total_home_score, total_away_score) ~ home_team + away_team,
                     data = finalScores22,
```

```
                      k = i)
  mses = append(mses, summary(elo22mod)$mse)
}
min(mses)
```

```
## [1] 0.2312386
```

```
which.min(mses) * 0.2
```

```
## [1] 52.6
```

After running this loop, I see that the smallest MSE for 2022 was 0.231 when K was 52.6. I will use this as my learning rate, and then fit the ELO model with results below. While not shown directly above, I did the same code for 2023, finding an MSE of 0.245 at k=24.8.

```
# Find percentage of time home teams wins (56.7)
hfaELO = as.numeric(finalScores %>%
  ungroup() %>%
  summarise(mean(home_win))*100)

# Fit ELO models for 2022 and 2023
elo22mod = elo.run(score(total_home_score, total_away_score) ~
                     adjust(home_team, hfaELO) + away_team,
                     data = finalScores22,
                     k = 52.6)
elo23mod = elo.run(score(total_home_score, total_away_score) ~
                     adjust(home_team, hfaELO) + away_team,
                     data = finalScores23,
                     k = 24.8)

# ELO Ratings by Team
final_elos22 = final.elos(elo22mod) %>%
  enframe() %>%
  arrange(desc(value))
final_elos23 = final.elos(elo23mod) %>%
  enframe() %>%
  arrange(desc(value))

kable(head(final_elos22, 10), caption = "Top 10 Teams by ELO 2022-2023 Season")
```

Table 8: Top 10 Teams by ELO 2022-2023 Season

name	value
KC	1735.883
CIN	1692.176
SF	1672.954
BUF	1655.774
PHI	1652.847
DAL	1605.961
MIN	1586.614

name	value
PIT	1567.552
JAX	1565.026
NYG	1543.732

```
kable(head(final_elos23, 10), caption = "Top 10 Teams by ELO 2023-2024 Season")
```

Table 9: Top 10 Teams by ELO 2023-2024 Season

name	value
KC	1589.274
BAL	1583.826
SF	1574.969
DET	1572.340
DAL	1553.343
BUF	1551.649
CLE	1538.849
LA	1533.403
HOU	1531.833
MIA	1527.959

In the ELO model with a home field advantage adjustment, we now see the Chiefs unsurprisingly take the top spot for both seasons. An interesting note is that in 2022-2023 season, the Eagles made it to the superbowl but are ranked fifth by ELO, although their rating is very close to all three teams above them.

4. Power Rankings

For the power ratings, I will use the dataset from the EPA per play, but split the data into two sets of special teams and non-special teams. The start of the data for non-special teams is seen below. As a note, the data is only rounded to fit nicer in the display, it was not rounded for the calculations.

```
# Start by filtering out the special teams, add home team column
prdata = epa_data %>%
  filter(special == 0) %>%
  select(game_id, season, home_team, away_team, posteam, defteam,
         epa_total, epa_play, plays) %>%
  mutate(team_home = ifelse(home_team == posteam, 1, -1))

# Data of only special teams
prdataSpecial = epa_data %>%
  filter(special == 1) %>%
  select(game_id, season, home_team, away_team, posteam, defteam,
         epa_total, epa_play, plays) %>%
  mutate(team_home = ifelse(home_team == posteam, 1, -1))

kable(head(prdata %>%
             mutate(across(where(is.numeric), round, digits = 2))),
      caption = "Sample of EPA Data No Special Teams")
```

Table 10: Sample of EPA Data No Special Teams

game_id	season	home_team	away_team	posteam	defteam	epa_total	epa_play	plays	team_home
2022_01_BAL_NYJ	2022	NYJ	BAL	BAL	NYJ	0.22	0.00	60	-1
2022_01_BAL_NYJ	2022	NYJ	BAL	NYJ	BAL	-14.76	-0.17	89	1
2022_01_BUF_LA	2022	LA	BUF	BUF	LA	8.51	0.13	64	-1
2022_01_BUF_LA	2022	LA	BUF	LA	BUF	-17.91	-0.24	76	1
2022_01_CLE_CAR	2022	CAR	CLE	CAR	CLE	0.59	0.01	63	1
2022_01_CLE_CAR	2022	CAR	CLE	CLE	CAR	4.43	0.05	89	-1

Now that we have the datasets, we can apply the stan_glmer function to our four different datasets to get the team effects for each season. The model fitting can be seen below.

```
# Building the stan models
stan_mod22 <- stan_glmer(epa_play ~ team_home + (1|posteam) + (1|defteam),
             data = prdata22,
             iter = 500,
             chains = 2,
             refresh = 0) # Use this line to hide the sampling details
stan_mod23 <- stan_glmer(epa_play ~ team_home + (1|posteam) + (1|defteam),
             data = prdata23,
             iter = 500,
             chains = 2,
             refresh = 0) # Use this line to hide the sampling details
stan_mod22Special <- stan_glmer(epa_play ~ team_home + (1|posteam) + (1|defteam),
             data = prdataSpecial22,
             iter = 500,
             chains = 2,
             refresh = 0) # Use this line to hide the sampling details
stan_mod23Special <- stan_glmer(epa_play ~ team_home + (1|posteam) + (1|defteam),
             data = prdataSpecial23,
             iter = 500,
             chains = 2,
             refresh = 0) # Use this line to hide the sampling details
```

Now that our models have been fit, we can get the overall team effects. I will normalize the data by multiplying the offensive and defensive effects (no special teams) by the average number of plays in the 2022 season (73.3) and the special teams effects by multiplying by the average number of special teams plays per game during the season (13.2). I will show the 2022 code to get the power rankings below, and details for 2023 (which are the same as 2022) can be found in the appendix at the end.

```
# Mean number of plays in 2022 to normalize
plays22 = as.numeric(prdata22 %>%
  summarise(mean(plays)))
plays22Special = as.numeric(prdataSpecial22 %>%
  summarise(mean(plays)))

# Get team effects without special teams
bayes_team_effects22 <- tibble(team = row.names(ranef(stan_mod22)$posteam),
                         bayes_off = ranef(stan_mod22)$posteam[,1]) %>%
  full_join(
    tibble(team = row.names(ranef(stan_mod22)$defteam),
```

```
            bayes_def = -ranef(stan_mod22)$defteam[,1]),
    by = c("team")
  ) %>%
  mutate(bayes_off = bayes_off*plays22,
         bayes_def = bayes_def*plays22)

bayes_special_effects22 <- tibble(team = row.names(ranef(stan_mod22Special)$posteam),
                          bayes_off_special = ranef(stan_mod22Special)$posteam[,1]) %>%
  full_join(
    tibble(team = row.names(ranef(stan_mod22Special)$defteam),
           bayes_def_special = -ranef(stan_mod22Special)$defteam[,1]),
    by = c("team")
  ) %>%
  mutate(bayes_off_special = bayes_off_special*plays22Special,
         bayes_def_special = bayes_def_special*plays22Special)

# Combining and getting our Bayes effects for 2022
bayes_effects22 = left_join(bayes_team_effects22, bayes_special_effects22, by = "team") %>%
  mutate(bayes_off = bayes_off + bayes_off_special,
         bayes_def = bayes_def + bayes_def_special,
         bayes_team = bayes_off + bayes_def,
         power_rank = min_rank(desc(bayes_team))) %>%
  arrange(power_rank) %>%
  select(team, bayes_off, bayes_def, bayes_team, power_rank)

kable(head(bayes_effects22 %>%
             mutate(across(where(is.numeric), round, digits = 3)), 10),
      caption = "Top 10 Teams By Power Ranking 2022-2023 Season")
```

Table 11: Top 10 Teams By Power Ranking 2022-2023 Season

team	bayes_off	bayes_def	bayes_team	power_rank
KC	8.867	-2.432	6.435	1
BUF	4.366	1.436	5.802	2
PHI	4.579	1.052	5.630	3
SF	3.181	2.301	5.482	4
CIN	4.176	0.157	4.333	5
DAL	2.243	1.679	3.922	6
DET	1.822	1.973	3.794	7
JAX	2.536	-0.579	1.956	8
MIA	2.000	-0.387	1.613	9
BAL	1.200	0.303	1.504	10

Table 12: Top 10 Teams By Power Ranking 2023-2024 Season

team	bayes_off	bayes_def	bayes_team	power_rank
SF	9.967	0.973	10.940	1
BAL	4.781	3.440	8.221	2
DAL	6.624	1.072	7.696	3
BUF	6.844	0.184	7.028	4

team	bayes_off	bayes_def	bayes_team	power_rank
GB	6.314	-1.147	5.167	5
DET	5.378	-0.372	5.006	6
KC	2.881	2.065	4.945	7
MIA	5.585	-0.683	4.902	8
LA	3.328	-0.096	3.232	9
NO	-0.105	1.933	1.828	10

The power rankings give the most interesting result, with the Chiefs being ranked 5th last year despite winning the Superbowl. Other teams such as San Francisco, Baltimore, and Buffalo all appear right at the top, which is unsurprising as these were the other teams in the conference championships. The team that is most surprising to see towards the top is the Dallas Cowboys, but after looking at their results it seems their high scoring offense helps pull them up. In 2022 the rankings were a bit more standard, with the four teams in the Conference Championships taking the top four spots.

5. Bayes Power Rankings

For fitting the Bayes Power Rankings, I references the power rankings example file and used the Stan file provided from class. Below is the code I used, where I altered the unique teams dataset and created my own Bayes dataset to fit the requirements for the Stan Data, and then ran the models for each of the seasons. I approached this the same as the power rankings above, where I fit the model for special teams and non-special teams, and then combined the information at the end for the cumulative ratings. Since I only had two seasons loaded for this project, I only did a Bayes Power Ranking for 2023-2024 season using the previous season's power rankings as my prior.

```
# Create our unique teams dataset with priors from previous season
unique_teams22 = bayes_team_effects22 %>%
  mutate(team_index = 1:n(),
         off_team_prior_mean = bayes_off,
         def_team_prior_mean = bayes_def
         ) %>%
  select(team, team_index, off_team_prior_mean, def_team_prior_mean)

# Create a dataset (I called bayesdata), which contains the team index for each team
bayesdata = left_join(prdata23, unique_teams22, by = join_by(posteam == team)) %>%
  rename(off_team_index = team_index)
bayesdata = left_join(bayesdata, unique_teams22, by = join_by(defteam == team)) %>%
  rename(def_team_index = team_index)

# Create our stan dataset, where 2023 EPA is our response, 2022 as prior
stan_data22 <- list(y = bayesdata$epa_play * plays23,
                    N = nrow(bayesdata),

                    hfa = bayesdata$team_home,

                    n_teams = nrow(unique_teams22),

                    off_team_prior_mean = unique_teams22$off_team_prior_mean,
                    def_team_prior_mean = unique_teams22$def_team_prior_mean,

                    off_team_index = bayesdata$off_team_index,
```

```
                  def_team_index = bayesdata$def_team_index
                  )

stan_with_prior_model22 <- rstan::stan(file = "power_rating_with_prior.stan",
                                       data = stan_data22,
                                       seed = 5,
                                       iter = 650,
                                       warmup =150,
                                       chains = 2,
                                       thin = 1,
                                       save_warmup = FALSE,
                                       refresh = 0)
```

Now that the models have been fit, I can clean the data and add together the special teams and non-special teams information to get the overall team estimates and rankings.

```
# Clean the data to get our Bayes Power Ranking for each team
bayes_w_prior_coefficients23 <- broom.mixed::tidy(stan_with_prior_model22)
bayes_team_effects23_priors <- bayes_w_prior_coefficients23 %>%
  filter(str_detect(term, "off|def")) %>%
  mutate(off_def = ifelse(str_detect(term, "off"), "off", "def"),
         team_index = str_extract(term, "[[:digit:]]+")) %>%
  select(-term) %>%
  pivot_wider(values_from = c('estimate', 'std.error'),
              names_from = 'off_def') %>%
  mutate(estimate_def = -estimate_def) %>%
  mutate(bayes_team_estimate = estimate_off + estimate_def) %>%
  mutate(team_index = as.numeric(team_index)) %>%
  left_join(unique_teams22 %>% mutate(team_index = as.numeric(team_index)),
            by = 'team_index')
bayes23ranks_priors = bayes_team_effects23_priors %>%
  select(team, bayes_team_estimate)

# Do the same for special teams ranks
bayes_w_prior_coefficients23Special <- broom.mixed::tidy(stan_with_prior_model22Special)
bayes_team_effects23_priorsSpecial <- bayes_w_prior_coefficients23Special %>%
  filter(str_detect(term, "off|def")) %>%
  mutate(off_def = ifelse(str_detect(term, "off"), "off", "def"),
         team_index = str_extract(term, "[[:digit:]]+")) %>%
  select(-term) %>%
  pivot_wider(values_from = c('estimate', 'std.error'),
              names_from = 'off_def') %>%
  mutate(estimate_def = -estimate_def) %>%
  mutate(bayes_team_estimate_special = estimate_off + estimate_def) %>%
  mutate(team_index = as.numeric(team_index)) %>%
  left_join(unique_teams22 %>% mutate(team_index = as.numeric(team_index)),
            by = 'team_index')
bayes23ranks_priorsSpecial = bayes_team_effects23_priorsSpecial %>%
  select(team, bayes_team_estimate_special)

bayes23ranks_priors = left_join(bayes23ranks_priors, bayes23ranks_priorsSpecial, by = "team") %>%
  mutate(bayes_estimate_prior = bayes_team_estimate + bayes_team_estimate_special,
         bayes_rank = min_rank(desc(bayes_estimate_prior))) %>%
```

```
  select(team, bayes_estimate_prior, bayes_rank) %>%
  arrange(bayes_rank)
```

Table 13: Top 10 Teams by Bayes Rankings for 2023 using 2022 as Prior

team	bayes_estimate_prior	bayes_rank
SF	14.229	1
BAL	12.430	2
BUF	10.174	3
KC	9.290	4
DAL	8.716	5
DET	6.400	6
GB	5.507	7
MIA	5.402	8
LA	3.476	9
NO	2.076	10

While I am unsure of how to interpret our power rank estimates here, the Bayes rankings seem to be a combination of the previous sections rankings. Here we see our super bowl winners ranked 4th, but again the same repeat teams as our top. I will discuss more about the rankings in the next section.

6. Comparing the Models

Now that all the models have been run, we can view the full rankings for each season. To start I will discuss the 2022-2023 rankings, with a summary table shown below arranged by the average of the 3 rankings.

Table 14: Comparison of Rankings Across Models for 2022-2023 Season

team	BTRating	BTrank	ELO_value	ELOrank	PR_value	power_rank
KC	29.698	1	1735.883	1	6.435	1
BUF	21.019	3	1655.774	4	5.802	2
PHI	25.350	2	1652.847	5	5.630	3
CIN	18.365	4	1692.176	2	4.333	5
SF	11.386	7	1672.954	3	5.482	4
DAL	13.071	6	1605.961	6	3.922	6
MIN	14.374	5	1586.614	7	0.069	17
NYG	7.802	8	1543.732	10	1.051	12
DET	5.161	13	1532.007	11	3.794	7
JAX	4.471	15	1565.026	9	1.956	8
BAL	6.646	9	1506.183	15	1.504	10
MIA	6.037	10	1474.083	19	1.613	9
PIT	5.669	11	1567.552	8	-1.300	20
GB	4.897	14	1508.302	14	0.687	13
LAC	3.662	18	1515.341	13	1.268	11
WAS	5.645	12	1518.620	12	-1.555	21
NE	4.351	16	1497.997	16	-0.925	19
SEA	3.281	21	1484.333	17	0.510	14

team	BTRating	BTrank	ELO_value	ELOrank	PR_value	power_rank
CLE	3.282	20	1471.695	20	-0.155	18
NYJ	3.787	17	1419.074	25	-1.794	22
ATL	2.315	25	1435.018	23	0.133	16
CAR	2.340	24	1479.783	18	-2.650	25
LV	1.493	28	1428.498	24	0.425	15
TB	3.360	19	1444.949	22	-3.245	27
NO	2.574	23	1462.413	21	-2.415	24
TEN	2.747	22	1410.542	26	-2.790	26
LA	1.345	29	1364.990	29	-2.317	23
CHI	1.698	26	1340.940	30	-3.746	28
IND	1.681	27	1393.786	27	-7.738	32
DEN	1.128	30	1390.843	28	-4.625	30
ARI	1.000	31	1319.812	32	-4.462	29
HOU	0.556	32	1322.273	31	-6.674	31

After running all of the models, the most consistent part of the analysis was the the Chiefs were ranked as the top team for the 2022-2023 season. Spots 2 through 5 varied a little, but in each of the models it was either the Bills, 49ers, Eagles, or Bengals. Across all three models, Houston was always ranked as either the worst or second worst team. The largest discrepancy seems to come with the Vikings, who were ranked 5th by Bradley Terry, 7th by ELO, but 17th by the Power Rankings. This is likely due to the fact that they had a fairly strong winning record, but their EPA per play was still very small. I will discuss more about reasons for differences after the looking at the results for the 2023-2024 season which is seen below.

Table 15: Comparison of Rankings Across Models for 2023-2024 Season

team	BTRating	BTrank	ELO_value	ELOrank	PR_value	power_rank	bayes_estimate_prior	bayes_rank
BAL	12.208	1	1583.826	2	8.221	2	12.430	2
SF	8.625	2	1574.969	3	10.940	1	14.229	1
KC	7.948	3	1589.274	1	4.945	7	9.290	4
DET	6.609	4	1572.340	4	5.006	6	6.400	6
BUF	4.628	8	1551.649	6	7.028	4	10.174	3
DAL	4.582	9	1553.343	5	7.696	3	8.716	5
LA	4.504	10	1533.403	8	3.232	9	3.476	9
MIA	3.755	14	1527.959	10	4.902	8	5.402	8
GB	2.839	17	1523.095	11	5.167	5	5.507	7
CLE	6.290	5	1538.849	7	-1.422	19	0.627	15
CIN	5.166	6	1517.931	14	0.562	14	1.057	13
HOU	4.216	11	1531.833	9	1.032	13	-1.764	19
PHI	3.999	12	1522.374	12	1.642	12	-0.347	16
TB	2.759	18	1512.967	15	1.734	11	1.394	11
PIT	5.077	7	1522.329	13	-1.487	20	-0.486	17
NO	2.107	21	1509.283	17	1.828	10	2.076	10
JAX	3.916	13	1497.871	19	0.398	15	0.841	14
SEA	3.692	15	1511.576	16	0.150	16	-1.979	20
IND	3.203	16	1502.917	18	-0.201	17	-2.896	22
MIN	1.775	22	1469.352	24	-0.424	18	1.180	12
LV	2.234	19	1491.242	21	-2.947	25	-2.399	21
CHI	1.445	25	1480.777	22	-1.589	21	-1.637	18
DEN	2.223	20	1492.148	20	-2.208	22	-4.740	25
TEN	1.610	24	1451.231	27	-2.619	24	-5.978	26

team	BTRating	BTrank	ELO_value	ELOrank	PR_value	power_rank	bayes_estimate_prior	bayes_rank
ATL	1.208	27	1459.633	26	-3.516	26	-4.078	24
LAC	1.136	28	1423.990	29	-2.441	23	-3.647	23
NYJ	1.642	23	1470.970	23	-7.906	31	-8.636	28
NYG	1.286	26	1461.269	25	-7.260	28	-8.427	27
ARI	1.000	29	1425.300	28	-5.033	27	-11.128	31
NE	0.751	30	1413.720	30	-8.599	32	-9.961	29
CAR	0.320	32	1374.664	32	-7.282	29	-10.560	30
WAS	0.687	31	1407.917	31	-7.632	30	-13.750	32

The 2023-2024 provided many more discrepancies than the previous season. The table is arranged by the average of the four rankings. The ELO model ranked Kansas City as the best team, which seemed fitting since they won the Superbowl. However, Bradley Terry, Power Rankings, and Bayes Power Rankings ranked them 3rd, 5th, and 4th respectively. All four of the models had a different team as the top and bottom teams, but New England, Washington, and Carolina typically made up the bottom 3.

The differences are to be expected based on the different ways the models calculate the rankings. Bradley Terry only considers wins and losses, so a 60-0 win is treated the same as a 17-16 win. ELO considers the score, and power rankings consider how well the team is consistently across every play. The Bayes Power rankings seemed to bring teams from the power rankings towards a ranking that was closer to how the team actually finished, because we used the previous season as a prior. For example, we knew from the previous season that the Chiefs were a top team, so our Bayes Power Rankings brought them from 5th up to 4th. The new model considered a team's performance this season as well as their previous season.

While none of the rankings are perfect, this project shows how there are many different methods to effectively rank sports teams. Some methods, such as Bradley Terry, appear to be more naive than others, but that does not make the rankings any less valid. There is always variability in sports, and there will likely never be a completely perfect ranking system that everyone agrees on.

Appendix

```
knitr::opts_chunk$set(echo = FALSE)
knitr::opts_chunk$set(fig.pos = "!h", fig.align = "center", fig.width = 5, fig.height = 4)
# Loading necessary packages
library(tidyverse)
library(nflfastR)
library(BradleyTerry2)
library(elo)
library(kableExtra)
library(knitr)
library(lme4)
library(grid)
library(gridExtra)
library(rstanarm)
# Loading in data
twentytwo = load_pbp(2022)
twentythree = load_pbp(2023)
data = rbind(twentytwo, twentythree)
# Create EPA dataset
epa_data = data %>%
  filter(!is.na(posteam)) %>%
  group_by(game_id, season, home_team, away_team,
           posteam, defteam, special) %>%
  mutate_at(vars(epa), ~ replace_na(., 0)) %>% # Change NA to 0
  summarise(epa_total = sum(epa),
            epa_play = mean(epa),
            plays = n()) %>%
  ungroup()
kable(head(epa_data %>%
             select(-season) %>%
             mutate(across(where(is.numeric), round, digits = 2))),
      caption = "First rows of EPA per Play Dataset")
finalScores = data %>%
  group_by(game_id) %>%
  select(game_id, season, home_team, away_team, total_home_score, total_away_score) %>%
  slice(n())
kable(head(finalScores), caption = "First rows of Final Scores dataset")
# First, add column to indicate home win, away win, or tie
finalScores = finalScores %>%
  mutate(home_win = ifelse(total_home_score > total_away_score, 1, 0),
         away_win = ifelse(total_away_score > total_home_score, 1, 0),
         tie = ifelse(total_home_score == total_away_score, 1, 0))

# Change dataset to get number of wins of each team against one another
wins = finalScores %>%
  group_by(home_team, away_team, season) %>%
  summarise(home_wins = sum(home_win), away_wins = sum(away_win), ties = sum(tie)) %>%
  ungroup(home_team, away_team)
kable(head(wins), caption = "First rows of Wins dataset")
# Changing to factors so model runs properly
wins$home_team = as.factor(wins$home_team)
wins$away_team = as.factor(wins$away_team)
```

```r
# Add ties as half a win
wins = wins %>%
  mutate(home_wins = home_wins + 0.5*ties,
         away_wins = away_wins + 0.5*ties)

# Separate seasons
wins2022 = wins %>%
  filter(season == 2022)
wins2023 = wins %>%
  filter(season == 2023)

# Fit Bradley Terry for 2022
bt22mod = BTm(cbind(home_wins, away_wins), home_team, away_team,
                  data = wins2022, id = "team")

# Fit Bradley Terry for 2023
bt23mod = BTm(cbind(home_wins, away_wins), home_team, away_team,
                  data = wins2023, id = "team")

summary(bt22mod)
# Add in information about home field advantage
wins2022$home_team <- data.frame(team = wins2022$home_team, at_home = 1)
wins2022$away_team <- data.frame(team = wins2022$away_team, at_home = -1)
wins2023$home_team <- data.frame(team = wins2023$home_team, at_home = 1)
wins2023$away_team <- data.frame(team = wins2023$away_team, at_home = -1)

# Fit the new Bradley Terry models
bt22mod2 <- update(bt22mod, formula = ~ team + at_home)
bt23mod2 <- update(bt23mod, formula = ~ team + at_home)
# Compare the Fit of both models
AIC22NoHome = AIC(bt22mod)
BIC22NoHome = BIC(bt22mod)
AIC22Home = AIC(bt22mod2)
BIC22Home = BIC(bt22mod2)
AIC23NoHome = AIC(bt23mod)
BIC23NoHome = BIC(bt23mod)
AIC23Home = AIC(bt23mod2)
BIC23Home = BIC(bt23mod2)

AICs = c(AIC22NoHome, AIC22Home, AIC23NoHome, AIC23Home)
BICs = c(BIC22NoHome, BIC22Home, BIC23NoHome, BIC23Home)
Season = c(2022, 2022, 2023, 2023)
Advantage = c("None", "Home Field", "None", "Home Field")
fits = data.frame(Season, Advantage, AICs, BICs)
kable(fits %>%
        mutate(across(where(is.numeric), round, digits = 2)),
      caption = "AIC and BIC by Model and Season")
# Get the rankings as a named vector for each season
btranks22 = c(exp(BTabilities(bt22mod))[,1])
btranks22 = data.frame(team = names(btranks22), BTRating = btranks22, row.names = NULL)
btranks22 = btranks22 %>%
  arrange(desc(BTRating))
btranks23 = c(exp(BTabilities(bt23mod))[,1])
```

```r
btranks23 = data.frame(team = names(btranks23), BTRating = btranks23, row.names = NULL)
btranks23 = btranks23 %>%
  arrange(desc(BTRating))

kable(head(btranks22 %>%
             mutate(across(where(is.numeric), round, digits = 2)), 10),
      caption = "Top 10 Teams in 2022-2023 season by Bradley Terry")
kable(head(btranks23 %>%
             mutate(across(where(is.numeric), round, digits = 2)), 10),
      caption = "Top 10 Teams in 2023-2024 season by Bradley Terry")
kable(head(finalScores %>%
             select(-season) %>%
             rename(home_score = total_home_score,
                    away_score = total_away_score)), caption = "Sample of Final Scores Data")
finalScores22 = finalScores %>%
  filter(season == 2022)
finalScores23 = finalScores %>%
  filter(season == 2023)
# Find the value of k with the smallest MSE
mses = c()
for(i in seq(1, 100, by = 0.2)){
  elo22mod = elo.run(score(total_home_score, total_away_score) ~ home_team + away_team,
                     data = finalScores22,
                     k = i)
  mses = append(mses, summary(elo22mod)$mse)
}
min(mses)
which.min(mses) * 0.2
# Find the value of k with the smallest MSE
mses = c()
for(i in seq(1, 100, by = 0.2)){
  elo23mod = elo.run(score(total_home_score, total_away_score) ~ home_team + away_team,
                     data = finalScores23,
                     k = i)
  mses = append(mses, summary(elo23mod)$mse)
}
# Output was MSE of 0.245 and k=24.8
# Find percentage of time home teams wins (56.7)
hfaELO = as.numeric(finalScores %>%
  ungroup() %>%
  summarise(mean(home_win))*100)

# Fit ELO models for 2022 and 2023
elo22mod = elo.run(score(total_home_score, total_away_score) ~
                     adjust(home_team, hfaELO) + away_team,
                     data = finalScores22,
                     k = 52.6)
elo23mod = elo.run(score(total_home_score, total_away_score) ~
                     adjust(home_team, hfaELO) + away_team,
                     data = finalScores23,
                     k = 24.8)

# ELO Ratings by Team
```

```r
final_elos22 = final.elos(elo22mod) %>%
  enframe() %>%
  arrange(desc(value))
final_elos23 = final.elos(elo23mod) %>%
  enframe() %>%
  arrange(desc(value))

kable(head(final_elos22, 10), caption = "Top 10 Teams by ELO 2022-2023 Season")
kable(head(final_elos23, 10), caption = "Top 10 Teams by ELO 2023-2024 Season")
# Start by filtering out the special teams, add home team column
prdata = epa_data %>%
  filter(special == 0) %>%
  select(game_id, season, home_team, away_team, posteam, defteam,
         epa_total, epa_play, plays) %>%
  mutate(team_home = ifelse(home_team == posteam, 1, -1))

# Data of only special teams
prdataSpecial = epa_data %>%
  filter(special == 1) %>%
  select(game_id, season, home_team, away_team, posteam, defteam,
         epa_total, epa_play, plays) %>%
  mutate(team_home = ifelse(home_team == posteam, 1, -1))

kable(head(prdata %>%
             mutate(across(where(is.numeric), round, digits = 2))),
      caption = "Sample of EPA Data No Special Teams")
# Filter data to different seasons
prdata22 = prdata %>%
  filter(season == 2022)
prdata23 = prdata %>%
  filter(season == 2023)
prdataSpecial22 = prdataSpecial %>%
  filter(season == 2022)
prdataSpecial23 = prdataSpecial %>%
  filter(season == 2023)
# Building the stan models
stan_mod22 <- stan_glmer(epa_play ~ team_home + (1|posteam) + (1|defteam),
             data = prdata22,
             iter = 500,
             chains = 2,
             refresh = 0) # Use this line to hide the sampling details
stan_mod23 <- stan_glmer(epa_play ~ team_home + (1|posteam) + (1|defteam),
             data = prdata23,
             iter = 500,
             chains = 2,
             refresh = 0) # Use this line to hide the sampling details
stan_mod22Special <- stan_glmer(epa_play ~ team_home + (1|posteam) + (1|defteam),
             data = prdataSpecial22,
             iter = 500,
             chains = 2,
             refresh = 0) # Use this line to hide the sampling details
stan_mod23Special <- stan_glmer(epa_play ~ team_home + (1|posteam) + (1|defteam),
             data = prdataSpecial23,
```

```
            iter = 500,
            chains = 2,
            refresh = 0) # Use this line to hide the sampling details
# Mean number of plays in 2022 to normalize
plays22 = as.numeric(prdata22 %>%
  summarise(mean(plays)))
plays22Special = as.numeric(prdataSpecial22 %>%
  summarise(mean(plays)))

# Get team effects without special teams
bayes_team_effects22 <- tibble(team = row.names(ranef(stan_mod22)$posteam),
                         bayes_off = ranef(stan_mod22)$posteam[,1]) %>%
  full_join(
    tibble(team = row.names(ranef(stan_mod22)$defteam),
           bayes_def = -ranef(stan_mod22)$defteam[,1]),
    by = c("team")
  ) %>%
  mutate(bayes_off = bayes_off*plays22,
         bayes_def = bayes_def*plays22)

bayes_special_effects22 <- tibble(team = row.names(ranef(stan_mod22Special)$posteam),
                         bayes_off_special = ranef(stan_mod22Special)$posteam[,1]) %>%
  full_join(
    tibble(team = row.names(ranef(stan_mod22Special)$defteam),
           bayes_def_special = -ranef(stan_mod22Special)$defteam[,1]),
    by = c("team")
  ) %>%
  mutate(bayes_off_special = bayes_off_special*plays22Special,
         bayes_def_special = bayes_def_special*plays22Special)

# Combining and getting our Bayes effects for 2022
bayes_effects22 = left_join(bayes_team_effects22, bayes_special_effects22, by = "team") %>%
  mutate(bayes_off = bayes_off + bayes_off_special,
         bayes_def = bayes_def + bayes_def_special,
         bayes_team = bayes_off + bayes_def,
         power_rank = min_rank(desc(bayes_team))) %>%
  arrange(power_rank) %>%
  select(team, bayes_off, bayes_def, bayes_team, power_rank)

kable(head(bayes_effects22 %>%
             mutate(across(where(is.numeric), round, digits = 3)), 10),
      caption = "Top 10 Teams By Power Ranking 2022-2023 Season")
# Mean number of plays in 2023 to normalize
plays23 = as.numeric(prdata23 %>%
  summarise(mean(plays)))
plays23Special = as.numeric(prdataSpecial23 %>%
  summarise(mean(plays)))

# Get team effects without special teams
bayes_team_effects23 <- tibble(team = row.names(ranef(stan_mod23)$posteam),
                         bayes_off = ranef(stan_mod23)$posteam[,1]) %>%
  full_join(
    tibble(team = row.names(ranef(stan_mod23)$defteam),
```

```
            bayes_def = -ranef(stan_mod23)$defteam[,1]),
    by = c("team")
  ) %>%
  mutate(bayes_off = bayes_off*plays23,
         bayes_def = bayes_def*plays23)

bayes_special_effects23 <- tibble(team = row.names(ranef(stan_mod23Special)$posteam),
                          bayes_off_special = ranef(stan_mod23Special)$posteam[,1]) %>%
  full_join(
    tibble(team = row.names(ranef(stan_mod23Special)$defteam),
           bayes_def_special = -ranef(stan_mod23Special)$defteam[,1]),
    by = c("team")
  ) %>%
  mutate(bayes_off_special = bayes_off_special*plays23Special,
         bayes_def_special = bayes_def_special*plays23Special)

# Combining and getting our Bayes effects for 2023
bayes_effects23 = left_join(bayes_team_effects23, bayes_special_effects23, by = "team") %>%
  mutate(bayes_off = bayes_off + bayes_off_special,
         bayes_def = bayes_def + bayes_def_special,
         bayes_team = bayes_off + bayes_def,
         power_rank = min_rank(desc(bayes_team))) %>%
  arrange(power_rank) %>%
  select(team, bayes_off, bayes_def, bayes_team, power_rank)

kable(head(bayes_effects23 %>%
             mutate(across(where(is.numeric), round, digits = 3)), 10),
      caption = "Top 10 Teams By Power Ranking 2023-2024 Season")
# Create our unique teams dataset with priors from previous season
unique_teams22 = bayes_team_effects22 %>%
  mutate(team_index = 1:n(),
         off_team_prior_mean = bayes_off,
         def_team_prior_mean = bayes_def
         ) %>%
  select(team, team_index, off_team_prior_mean, def_team_prior_mean)

# Create a dataset (I called bayesdata), which contains the team index for each team
bayesdata = left_join(prdata23, unique_teams22, by = join_by(posteam == team)) %>%
  rename(off_team_index = team_index)
bayesdata = left_join(bayesdata, unique_teams22, by = join_by(defteam == team)) %>%
  rename(def_team_index = team_index)

# Create our stan dataset, where 2023 EPA is our response, 2022 as prior
stan_data22 <- list(y = bayesdata$epa_play * plays23,
                    N = nrow(bayesdata),

                    hfa = bayesdata$team_home,

                    n_teams = nrow(unique_teams22),

                    off_team_prior_mean = unique_teams22$off_team_prior_mean,
                    def_team_prior_mean = unique_teams22$def_team_prior_mean,
```

```
                  off_team_index = bayesdata$off_team_index,
                  def_team_index = bayesdata$def_team_index
                  )

stan_with_prior_model22 <- rstan::stan(file = "power_rating_with_prior.stan",
                                       data = stan_data22,
                                       seed = 5,
                                       iter = 650,
                                       warmup =150,
                                       chains = 2,
                                       thin = 1,
                                       save_warmup = FALSE,
                                       refresh = 0)
# Now, same analysis as above but with special teams
unique_teams22Special = bayes_special_effects22 %>%
  mutate(team_index = 1:n(),
         off_team_prior_mean = bayes_off_special,
         def_team_prior_mean = bayes_def_special
         ) %>%
  select(team, team_index, off_team_prior_mean, def_team_prior_mean)

# Create a dataset (I called bayesdataSpecial), which contains the team index for each team
bayesdataSpecial = left_join(prdata23, unique_teams22Special, by = join_by(posteam == team)) %>%
  rename(off_team_index = team_index)
bayesdataSpecial = left_join(bayesdataSpecial, unique_teams22Special,
                             by = join_by(defteam == team)) %>%
  rename(def_team_index = team_index)

# Create our stan dataset, where 2023 EPA is our response, 2022 as prior
stan_data22Special <- list(y = bayesdataSpecial$epa_play*plays23Special,
                  N = nrow(bayesdataSpecial),

                  hfa = bayesdataSpecial$team_home,

                  n_teams = nrow(unique_teams22Special),

                  off_team_prior_mean = unique_teams22Special$off_team_prior_mean,
                  def_team_prior_mean = unique_teams22Special$def_team_prior_mean,

                  off_team_index = bayesdataSpecial$off_team_index,
                  def_team_index = bayesdataSpecial$def_team_index
                  )

stan_with_prior_model22Special <- rstan::stan(file = "power_rating_with_prior.stan",
                                       data = stan_data22Special,
                                       seed = 5,
                                       iter = 1000,
                                       warmup =150,
                                       chains = 2,
                                       thin = 1,
                                       save_warmup = FALSE,
                                       refresh = 0)
# Clean the data to get our Bayes Power Ranking for each team
```

```r
bayes_w_prior_coefficients23 <- broom.mixed::tidy(stan_with_prior_model22)
bayes_team_effects23_priors <- bayes_w_prior_coefficients23 %>%
  filter(str_detect(term, "off|def")) %>%
  mutate(off_def = ifelse(str_detect(term, "off"), "off", "def"),
         team_index = str_extract(term, "[[:digit:]]+")) %>%
  select(-term) %>%
  pivot_wider(values_from = c('estimate', 'std.error'),
              names_from = 'off_def') %>%
  mutate(estimate_def = -estimate_def) %>%
  mutate(bayes_team_estimate = estimate_off + estimate_def) %>%
  mutate(team_index = as.numeric(team_index)) %>%
  left_join(unique_teams22 %>% mutate(team_index = as.numeric(team_index)),
            by = 'team_index')
bayes23ranks_priors = bayes_team_effects23_priors %>%
  select(team, bayes_team_estimate)

# Do the same for special teams ranks
bayes_w_prior_coefficients23Special <- broom.mixed::tidy(stan_with_prior_model22Special)
bayes_team_effects23_priorsSpecial <- bayes_w_prior_coefficients23Special %>%
  filter(str_detect(term, "off|def")) %>%
  mutate(off_def = ifelse(str_detect(term, "off"), "off", "def"),
         team_index = str_extract(term, "[[:digit:]]+")) %>%
  select(-term) %>%
  pivot_wider(values_from = c('estimate', 'std.error'),
              names_from = 'off_def') %>%
  mutate(estimate_def = -estimate_def) %>%
  mutate(bayes_team_estimate_special = estimate_off + estimate_def) %>%
  mutate(team_index = as.numeric(team_index)) %>%
  left_join(unique_teams22 %>% mutate(team_index = as.numeric(team_index)),
            by = 'team_index')
bayes23ranks_priorsSpecial = bayes_team_effects23_priorsSpecial %>%
  select(team, bayes_team_estimate_special)

bayes23ranks_priors = left_join(bayes23ranks_priors, bayes23ranks_priorsSpecial, by = "team") %>%
  mutate(bayes_estimate_prior = bayes_team_estimate + bayes_team_estimate_special,
         bayes_rank = min_rank(desc(bayes_estimate_prior))) %>%
  select(team, bayes_estimate_prior, bayes_rank) %>%
  arrange(bayes_rank)
kable(head(bayes23ranks_priors %>%
            mutate(across(where(is.numeric), round, digits = 3)), 10),
            caption = "Top 10 Teams by Bayes Rankings for 2023 using 2022 as Prior")
btranks22 = btranks22 %>%
  mutate(BTrank = row_number())
btranks23 = btranks23 %>%
  mutate(BTrank = row_number())
final_elos22 = final_elos22 %>%
  mutate(ELOrank = row_number(),
         team = name)
final_elos23 = final_elos23 %>%
  mutate(ELOrank = row_number(),
         team = name)
bayes_effects22 = bayes_effects22 %>%
  mutate(PowerRank = row_number())
```

```r
bayes_effects23 = bayes_effects23 %>%
  mutate(PowerRank = row_number())

# Make the rankings in one dataset
ranks22 <- list(btranks22, final_elos22, bayes_effects22)

# Using Reduce to join all data frames by 'id'
ranks22 <- Reduce(function(x, y) left_join(x, y, by = "team"), ranks22)

ranks22 = ranks22 %>%
  mutate(ELO_value = value, PR_value = bayes_team) %>%
  select(team, BTRating, BTrank, ELO_value, ELOrank, PR_value, power_rank)

# Now for 2023
ranks23 <- list(btranks23, final_elos23, bayes_effects23, bayes23ranks_priors)

# Using Reduce to join all data frames by 'id'
ranks23 <- Reduce(function(x, y) left_join(x, y, by = "team"), ranks23)

ranks23 = ranks23 %>%
  mutate(ELO_value = value, PR_value = bayes_team) %>%
  select(team, BTRating, BTrank, ELO_value, ELOrank, PR_value, power_rank,
         bayes_estimate_prior, bayes_rank)
ranks22 <- ranks22 %>%
  mutate(across(where(is.numeric), round, digits = 3))
kable(ranks22 %>%
  arrange((BTrank + ELOrank + power_rank)/3),
  caption = "Comparison of Rankings Across Models for 2022-2023 Season")
ranks23 <- ranks23 %>%
  mutate(across(where(is.numeric), round, digits = 3))
kable(ranks23 %>%
  arrange((BTrank + ELOrank + power_rank + bayes_rank)/4),
  caption = "Comparison of Rankings Across Models for 2023-2024 Season")
```