Multivariable Linear Regression

NCAA Men's Basketball Power Scores

We are given a dataset of the game results of the 2022-2023 NCAA men's basketball season,

Season	WLoc	WTeamName	LTeamName	ScoreDiff	WScore	LScore
<dbl>	<chr>	<chr>	<chr>	<dbl>	<dbl>	<dbl>
2023	H	DePaul	Loyola MD	6	72	66
2023	H	Duke	Jacksonville	27	71	44
2023	A	Evansville	Miami OH	-4	78	74
2023	A	FL Gulf Coast	USC	-13	74	61
2023	H	Florida	Stony Brook	36	81	45
2023	H	Florida Intl	Houston Chr	11	77	66

row = a game
i = index of i^{th} game
$H(i)$ = index of home team
$A(i)$ = index of away team
y_i = score differential of game i
(home score minus away score)

Supposing each team j has a latent (unobserved) power rating β_j, we model the outcome (score diff) of the i^{th} game by

$$y_i = \beta_0 + \beta_{H(i)} - \beta_{A(i)} + \varepsilon_i .$$

ε_i is mean zero noise, $\mathbb{E}[\varepsilon_i] = 0$.

What does β_0 represent?

$$Y_1 = \beta_0 + \beta_{DePaul} - \beta_{Loyola} + \varepsilon_1$$
$$Y_2 = \beta_0 + \beta_{Duke} - \beta_{Jacksonville} + \varepsilon_2$$
$$Y_3 = \beta_0 + \beta_{Miami} - \beta_{Evansville} + \varepsilon_3$$
$$\vdots$$

In Matrix-Vector Form:

$$\underbrace{\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \end{bmatrix}}_{\vec{y}} = \underbrace{\begin{array}{c} \begin{array}{ccccccc} \text{intercept} & \text{DePaul} & \text{Loyola} & \text{Duke} & \text{Jacksonville} & \text{Miami} & \text{Evansville} \end{array} \\ \begin{bmatrix} 1 & 1 & -1 & 0 & 0 & 0 & 0 & \cdots \\ 1 & 0 & 0 & 1 & -1 & 0 & 0 & \cdots \\ 1 & 0 & 0 & 0 & 0 & 1 & -1 & \\ & & & \vdots & & & & \end{bmatrix} \end{array}}_{X} \underbrace{\begin{bmatrix} \beta_0 \\ \beta_{DePaul} \\ \beta_{Loyola} \\ \beta_{Duke} \\ \beta_{Jacksonville} \\ \vdots \end{bmatrix}}_{\vec{\beta}} + \underbrace{\begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \varepsilon_3 \\ \vdots \end{bmatrix}}_{\vec{\varepsilon}}$$

Model $\vec{y} = X\vec{\beta} + \vec{\varepsilon} \longrightarrow y = X\beta + \varepsilon$

X = Scheduling matrix,

$j=1$, $X_{i1} = 1$ (intercept term)

$$j>1,\quad X_{ij} = X[\text{Row } i, \text{ column } j] = \begin{cases} 1 & \text{if home team in game } i \text{ is team } j-1 \\ -1 & \text{if away team in game } i \text{ is team } j-1 \\ 0 & \text{else} \end{cases}$$

```
> df_ncaamb2[1:5,]
# A tibble: 5 × 7
  Season WTeamName    LTeamName    WScore LScore WLoc  ScoreDiff
   <dbl> <chr>        <chr>         <dbl>  <dbl> <chr>     <dbl>
1   2023 Abilene Chr  Jackson St       65     56 H             9
2   2023 Akron        S Dakota St      81     80 H             1
3   2023 Alabama      Longwood         75     54 H            21
4   2023 Arizona      Nicholls St     117     75 H            42
5   2023 Arizona St   Tarleton St      62     59 H             3
> X[1:5,c(1:5,131)]
     (Intercept) Abilene Chr Air Force Akron Alabama Jackson St
[1,]           1           1         0     0       0         -1
[2,]           1           0         0     1       0          0
[3,]           1           0         0     0       1          0
[4,]           1           0         0     0       0          0
[5,]           1           0         0     0       0          0
```

How do we estimate the coefficients (e.g., the power ratings) β from observed data (X, y)?

Recall that in simple linear regression, we estimated (β_0, β_1) by minimizing the Residual Sum of Squares. Similarly, in multivariable linear regression we minimize the RSS,

$$\hat{\beta} = \underset{\beta}{\operatorname{argmin}}\ RSS(\beta) = \underset{\beta}{\operatorname{argmin}} \sum_{i=1}^{n} \left(y_i - x_i^T \beta\right)^2$$

where x_i is the i^{th} Row of X and

$$x_i^T \beta = x_i \cdot \beta = x_{i1}\beta_0 + x_{i2}\beta_1 + \dots + x_{i(k+1)}\beta_k$$

is the dot product.

Multivariable Calculus: set the gradient equal to 0. The gradient is the analog of the derivative.

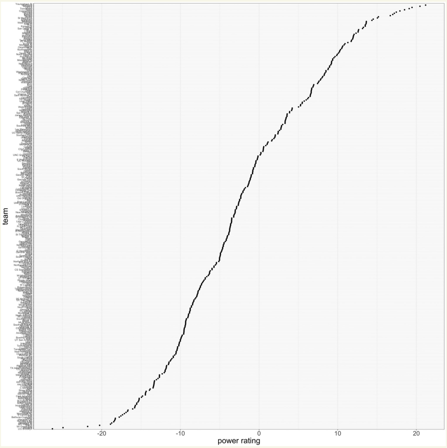
* Set $\nabla_\beta RSS(\beta) = 0$ and solve for β to obtain our estimate $\hat{\beta}$. $\boxed{\hat{\beta} = (X^T X)^{-1} X^T y}$ (see endnotes)

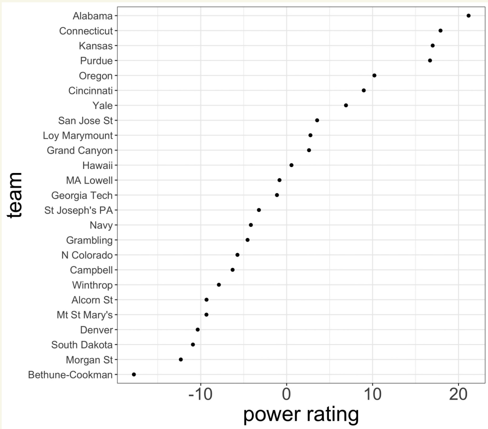
So, for our NCAA Basketball power Ratings model $y = X\beta + \varepsilon$, we now know how to estimate $\hat{\beta}$. Let's run the computation and see what it says!

```
### get power ratings using multivariable linear regression
power_ratings_model = lm(df_ncaamb2$ScoreDiff ~ X + 0)
power_ratings = power_ratings_model$coefficients
```

Intercept $\hat{\beta}_0 = 2$ → Home Court Advantage!

Too many teams to see. Some power ratings:






```
> tibble(teams=colnames(X), power_ratings=unname(power_ratings)) %>%
+   drop_na() %>%
+   arrange(power_ratings) %>%
+   head(5)
# A tibble: 5 × 2
  teams         power_ratings
  <chr>                 <dbl>
1 LIU Brooklyn          -26.3
2 Hartford              -24.9
3 WI Green Bay          -21.8
4 IUPUI                 -20.3
5 MS Valley St          -18.9
> tibble(teams=colnames(X), power_ratings=unname(power_ratings)) %>%
+   drop_na() %>%
+   arrange(-power_ratings) %>%
+   head(5)
# A tibble: 5 × 2
  teams     power_ratings
  <chr>             <dbl>
1 Alabama            21.2
2 Houston            20.5
3 UCLA               19.4
4 Tennessee          19.1
5 Texas              18.5
```

Expected Points in American Football

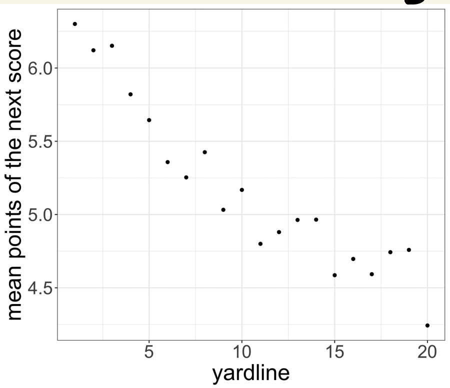
We are given a dataset of NFL plays,

row = a play
i = index of i^{th} play
x_i = yardline (yards from opponent's endzone)
y_i = net points of the next score in the half $\in \{7, 3, 2, 0, -2, -3, -7\}$

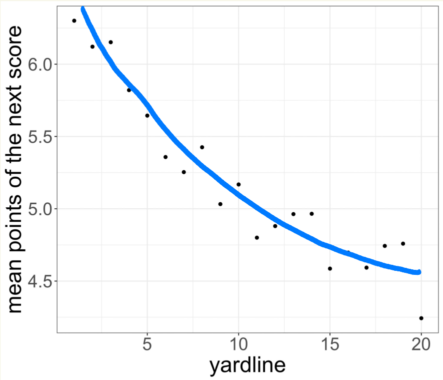
We model $y_i = f(ydl_i) + \varepsilon_i$ for some function f.

$\mathbb{E}[y_i] = f(ydl_i)$ <u>expected points</u> and we want to estimate this quantity.

Generally, it is smart to begin with plotting:



The Relationship looks quadratic, not linear:



Begin with just the redzone for simplicity.

How can we use linear Regression to capture a nonlinear Relationship?? Data transformations!

Linear Model $Y_i = \beta_0 + \beta_1 X_i + \varepsilon_i$ $\qquad \mathbb{E}\,\varepsilon_i = 0$ (mean zero noise)

Quadratic Model $Y_i = \beta_0 + \beta_1 X_i + \beta_2 X_i^2 + \varepsilon_i$

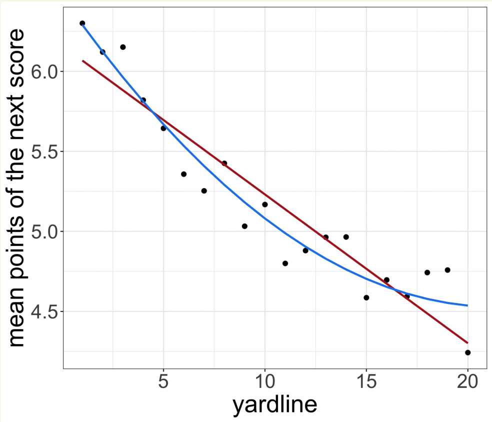
In matrix vector form:

$$\begin{bmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_n \end{bmatrix} = \begin{bmatrix} 1 & x_1 & x_1^2 \\ 1 & x_2 & x_2^2 \\ & \vdots & \\ 1 & x_n & x_n^2 \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \end{bmatrix} + \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \varepsilon_3 \end{bmatrix}$$

$$\rightarrow y = X^T \beta + \varepsilon$$

Estimate the coefficients as before, $\hat{\beta} = (X^T X)^{-1} X^T y$.

```
> m_ep_linear = lm(data=D3r, pts_next_score ~ yardline_100)
> m_ep_linear

Call:
lm(formula = pts_next_score ~ yardline_100, data = D3r)

Coefficients:
 (Intercept)  yardline_100
     6.16098      -0.09299

> ### quadratic model
> m_ep_quad = lm(data=D3r, pts_next_score ~ yardline_100 + I(yardline_100^2))
> m_ep_quad

Call:
lm(formula = pts_next_score ~ yardline_100 + I(yardline_100^2),
    data = D3r)

Coefficients:
      (Intercept)       yardline_100  I(yardline_100^2)
         6.467712          -0.180798           0.004212
```



Quadratic model looks better!

NFL Draft Expected Value Curve

We are given a dataset of NFL draft picks,

- row = a draft pick
- i = index of i^{th} draft pick
- x_i = player i's draft pick number
- y_i = player i's first contract "performance value"

player_id	player_name	year	t	draft_pos	firstContractPerformanceValue
40688	A.J. Bouye	2013	1	NA	1.659893e-02
42410	A.J. Cann	2015	1	67	3.541377e-02
35558	A.J. Edds	2011	1	119	-7.510774e-04
37077	A.J. Green	2011	1	4	8.018761e-02
30819	A.J. Hawk	2006	1	5	4.689117e-02
35863	A.J. Jefferson	2010	1	NA	4.419914e-03
38560	A.J. Jenkins	2012	1	30	5.734494e-03
40096	A.J. Klein	2013	1	148	1.305431e-02
30972	A.J. Nicholson	2006	1	157	-2.287106e-03

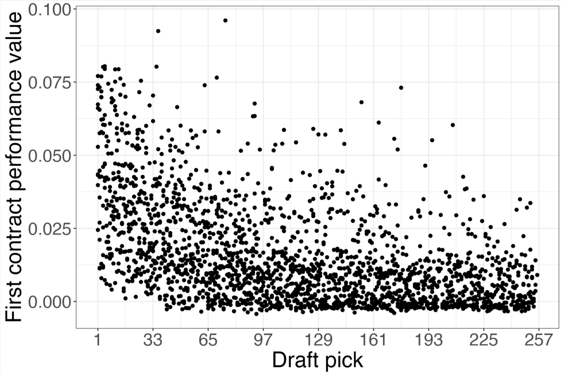
Think of first contract value as second contract compensation (assume a relatively efficient market) as a percentage of the salary cap.
I think in this dataset, we instead use First Contract Performance Value from Massey Thaler (2013).

We model the outcome of a draft pick by

$$y_i = f(x_i) + \varepsilon_i$$

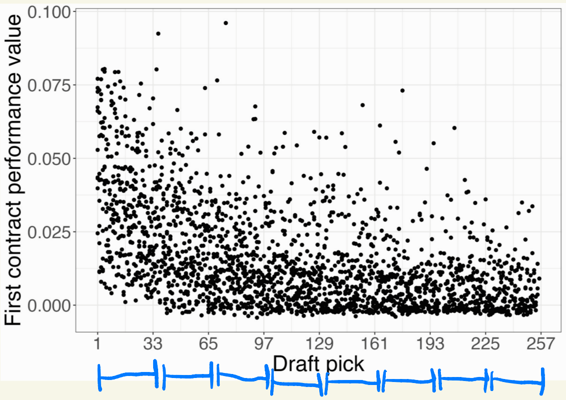
and want to estimate the expected value curve $x \mapsto f(x)$.

EDA: value vs. draft pick



The expected value $E(y|x)$ is nonlinear!
In particular, it's convex:
The dropoff in value b/t picks t and $t+1$ decreases as t increases.

To model a general nonlinear shape, use a $\boxed{\text{spline}}$.
To fit a spline, you fit a separate polynomial (usually a cubic) to different subsections of the data.



For instance, imagine fitting a separate cubic in each Round of the draft

These separators (e.g. $x = 33, 65, \ldots, 225$) are called Knots.

To force the fitted spline to be smooth, we mandate that at each knot the curve has
the same left y value and right y value,
the same left derivative and right derivative,
and the same left 2^{nd} derivative and right 2^{nd} derivative.

Suppose we fit a cubic spline with one knot at $x=k$
(e.g. $x=129$, middle of the draft)

We model $y_i = f(x_i|\beta) + \varepsilon_i$
where f is the spline and β are the spline parameters,

$$f(x|\beta) = \begin{cases} \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 & \text{if } x \le k \\ \beta_4 + \beta_5 x + \beta_6 x^2 + \beta_7 x^3 & \text{if } x \ge k \end{cases}$$

Enforce
$$\begin{cases} \lim_{x \to k^-} f(x|\beta) = \lim_{x \to k^+} f(x|\beta) \\ \lim_{x \to k^-} f'(x|\beta) = \lim_{x \to k^+} f'(x|\beta) \\ \lim_{x \to k^-} f''(x|\beta) = \lim_{x \to k^+} f''(x|\beta) \end{cases}$$

$$\begin{cases} \beta_0 + \beta_1 K + \beta_2 K^2 + \beta_3 K^3 = \beta_4 + \beta_5 K + \beta_6 K^2 + \beta_7 K^3 \\ \beta_1 + 2\beta_2 K + 3\beta_3 K^2 = \beta_5 + 2\beta_6 K + 3\beta_7 K^2 \\ 2\beta_2 + 6\beta_3 K = 2\beta_6 + 6\beta_7 K \end{cases}$$

$$\begin{cases} \beta_7 = (2\beta_2 + 6\beta_3 K - 2\beta_6) / (6K) \\ \beta_6 = (\beta_1 + 2\beta_2 K + 3\beta_3 K^2 - \beta_5 - 3\beta_7 K^2) / (2K) \\ \beta_5 = (\beta_0 + \beta_1 K + \beta_2 K^2 + \beta_3 K^3 - \beta_4 - \beta_6 K^2 - \beta_7 K^3) / (K) \end{cases}$$

$\beta_5, \beta_6, \beta_7$ are completely determined by $\beta_0, \beta_1, \beta_2, \beta_3, \beta_4$

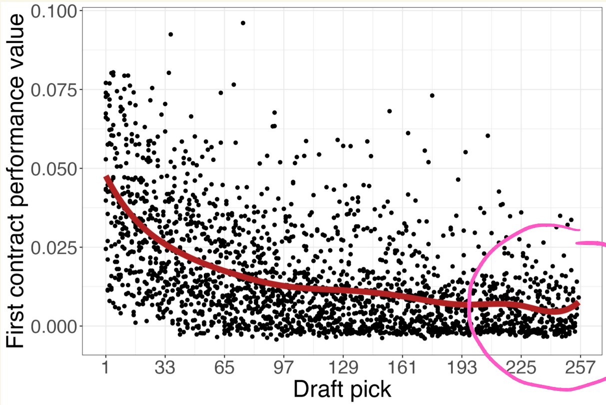
So we only need to estimate 5 parameters!

Parameters then estimated by forming the model matrix and performing a multivariable regression.

* Let's fit the spline model.

Here the knots are the start of each Round:

```
draft_model1 = lm(
  firstContractPerformanceValue ~ splines::bs(draft_pos, degree=3, knots=seq(33,32*8,by=32)),
  data=df_draft
)
draft_model1
```



a bit wiggly at the end; too many knots, not enough data

K knots & cubic spline → K+3+1 degrees of freedom

"df=5" → 1 auto-set knot (equally spaced)

```
draft_model2 = lm(
  firstContractPerformanceValue ~ splines::bs(draft_pos, degree=3, df=5),
  data=df_draft
)
draft_model2
```