WSABI Summer Research Lab **Summer 2024**

Lecture 2: Multivariable Linear Regression

Instructor: Ryan Brill *Scribe: Jonathan Pipping*

2.1 Motivating Example: NCAA Men's Basketball Power Ratings

We have a dataset of game results from the 2022-2023 NCAA men's basketball season. As pictured in Figure 2.1, each row represents a game and includes the following information:

- i: index of the i^{th} game
- $H(i)$: index of the home team
- $A(i)$: index of the away team
- y_i: the score differential of game i (home team score - away team score)

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

Figure 2.1: Head of the 2022-2023 NCAA men's basketball dataset

We intend to use this data to generate power ratings (team strength) for each team, which we can do by setting up a model for the score differential.

2.2 Setting Up the Model

Suppose each team j has a latent (unobserved) power rating β_j. Then, we can model the outcome (score differential) of the i^{th} game as follows:

$$y_i = \beta_0 + \beta_{H(i)} - \beta_{A(i)} + \epsilon_i \tag{2.1}$$

where $\beta_{H(i)}$ and $\beta_{A(i)}$ are unknown constants representing the strength of the home and away teams, and ϵ_i is a mean-zero noise term ($\mathbb{E}[\epsilon_i] = 0$). What does β_0 represent?

What does this look like in each line of the dataset?

$$\begin{aligned} y_1 &= \beta_0 + \beta_{DePaul} - \beta_{Loyola} + \epsilon_1 \\ y_2 &= \beta_0 + \beta_{Duke} - \beta_{Jacksonville} + \epsilon_2 \\ y_3 &= \beta_0 + \beta_{Miami} - \beta_{Evansville} + \epsilon_3 \\ &\vdots \end{aligned}$$

In matrix-vector form, we can write this as

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \end{bmatrix} = \begin{bmatrix} 1 & 1 & -1 & 0 & 0 & 0 & 0 & \dots \\ 1 & 0 & 0 & 1 & -1 & 0 & 0 & \dots \\ 1 & 0 & 0 & 0 & 0 & 1 & -1 & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_{DePaul} \\ \beta_{Loyola} \\ \beta_{Duke} \\ \beta_{Jacksonville} \\ \beta_{Miami} \\ \beta_{Evansville} \\ \vdots \end{bmatrix} + \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \epsilon_3 \\ \vdots \end{bmatrix} \tag{2.2}$$

We can write this more compactly as

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon} \tag{2.3}$$

where $\mathbf{y}$ is a vector of score differentials, $\mathbf{X}$ is a scheduling matrix, $\boldsymbol{\beta}$ is a vector of unknown parameters, and $\boldsymbol{\epsilon}$ is a vector of noise terms. Note that $\mathbf{X}$ consists of an intercept column and a one-hot encoding of the home (1) and away (-1) teams in the remaining columns.

Now, how do we estimate the unknown team strength parameters $\boldsymbol{\beta}$ from the observed data $(\mathbf{X}, \mathbf{y})$?

2.3 Estimating Model Parameters

Recall that in Lecture 1 we estimated (β_0, β_1) by minimizing the **Residual Sum of Squares**. Similarly, in multi-variable regression, we minimize the RSS. The optimization problem is:

$$\begin{aligned} \widehat{\boldsymbol{\beta}} &= \arg\min_{\boldsymbol{\beta}} \text{RSS}(\boldsymbol{\beta}) \\ &= \arg\min_{\boldsymbol{\beta}} \sum_{i=1}^{n} (y_i - x_i^T \boldsymbol{\beta})^2 \end{aligned} \tag{2.4}$$

where x_i is the i^{th} row of $\mathbf{X}$ and $x_i^T\boldsymbol{\beta} = x_i \cdot \boldsymbol{\beta} = x_{i1}\beta_0 + x_{i2}\beta_1 + \ldots + x_{i(j+1)}\beta_j$ is the dot product.

We can solve this with calculus, setting the gradient (the multivariable analog of the derivative) equal to 0 and solving for $\boldsymbol{\beta}$ to obtain our estimate $\widehat{\boldsymbol{\beta}}$. Our solution to this problem is:

$$\widehat{\boldsymbol{\beta}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y} \tag{2.5}$$

These are called the **Normal Equations** for linear regression.

2.4 Back to Our Power Ratings Model

Now that we know how to estimate $\widehat{\boldsymbol{\beta}}$, we can use `R` to fit our regression model.

```
# fit the linear model
model = lm(score_diff ~ X + 0, data = ncaa_data)
ratings = model$coefficients
```

Our intercept $\widehat{\beta}_0$ is the average score differential, and since `score_diff` is the difference between the home and away score, $\widehat{\beta}_0$ is the home field advantage. In our model, $\widehat{\beta}_0 = 2$ means that being the home team is associated with a 2-point scoring advantage.

We plot the sorted power ratings for each team in Figure 2.2. A zoom-in on 25 randomly-selected teams is shown in Figure 2.3. If we wanted to predict the score differential for a game between two teams, we would simply take the difference between their power ratings and add the home field advantage.

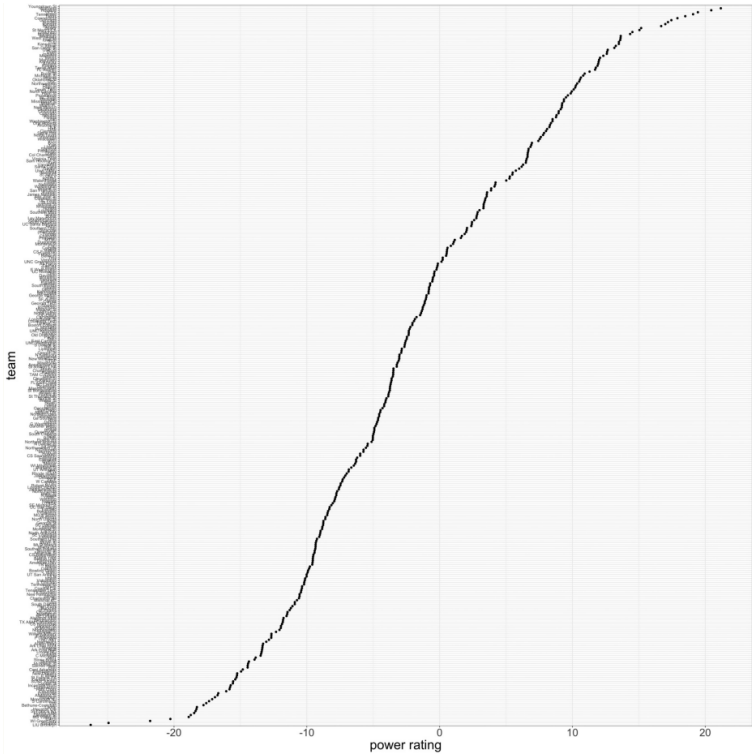

Figure 2.2: Team power ratings for the 2022-2023 NCAA men's basketball season

2.5 Example: Expected Points in American Football

We have a dataset of play-by-play data from the 2022-2023 NFL season. Each row represents a play and includes the following information:

- i: index of the i^{th} play
- X_i: yard line (yards from opponent's end zone)
- y_i: net points of the next score in the half $\in \{7, 3, 2, 0, -2, -3, -7\}$

We model y_i as a function of X_i. Mathematically, we express this as

$$y_i = f(X_i) + \epsilon_i \tag{2.6}$$

where f is some unknown function and ϵ_i is a mean-zero noise term. As in Section 2.5, we want to estimate the expected value of this quantity:

$$\mathbb{E}[y_i \mid X_i] = f(X_i) \tag{2.7}$$

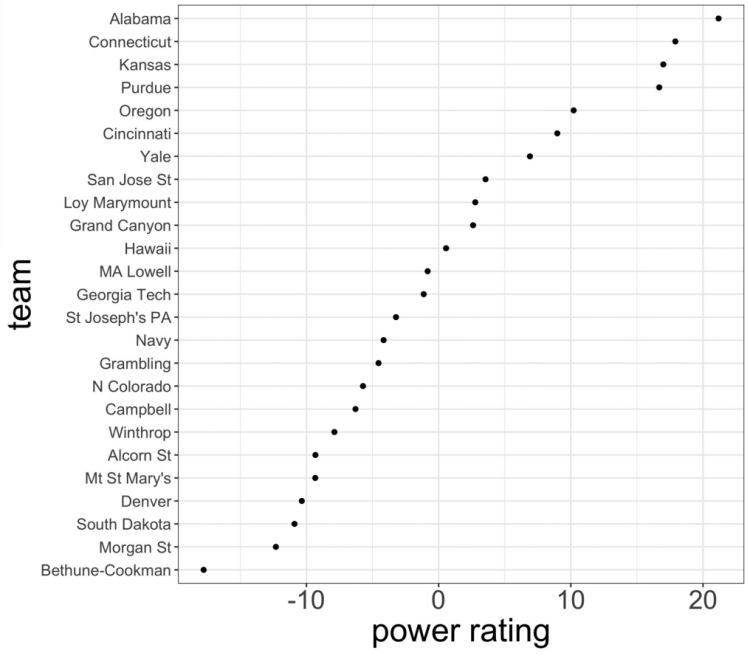

Figure 2.3: Zoom-in on 25 randomly-selected teams in the 2022-2023 NCAA men's basketball season

Generally, it's a good idea to start with some exploratory data analysis. We plot the relationship between yard line (X_i) and net points (y_i) in Figure 2.4. As highlighted in Figure 2.5, we see that the relationship bewteen yard line and net points looks quadratic, not linear. How can we use linear regression to capture nonlinear relationships?

2.6 Data Transformations

We can use a **data transformation** to capture nonlinear relationships. For example, we can transform the yard line X_i to X_i^2 to capture the quadratic relationship we noted in Figure 2.5.

If our linear model setup would be

$$y_i = \beta_0 + \beta_1 X_i + \epsilon_i, \quad \mathbb{E}[\epsilon_i] = 0 \tag{2.8}$$

then our transformed **quadratic model** is

$$y_i = \beta_0 + \beta_1 X_i + \beta_2 X_i^2 + \epsilon_i, \quad \mathbb{E}[\epsilon_i] = 0 \tag{2.9}$$

In matrix-vector form, we maintain the form

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon} \tag{2.10}$$

but now $\mathbf{X}$ is a matrix with three columns instead of two: the intercept column, the X_i column, and the X_i^2 column. Explicitly, we have

$$\underbrace{\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \end{bmatrix}}_{\mathbf{y}} = \underbrace{\begin{bmatrix} 1 & X_1 & X_1^2 \\ 1 & X_2 & X_2^2 \\ 1 & X_3 & X_3^2 \\ \vdots & \vdots & \vdots \end{bmatrix}}_{\mathbf{X}} \underbrace{\begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \end{bmatrix}}_{\boldsymbol{\beta}} + \underbrace{\begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \epsilon_3 \\ \vdots \end{bmatrix}}_{\boldsymbol{\epsilon}} \tag{2.11}$$

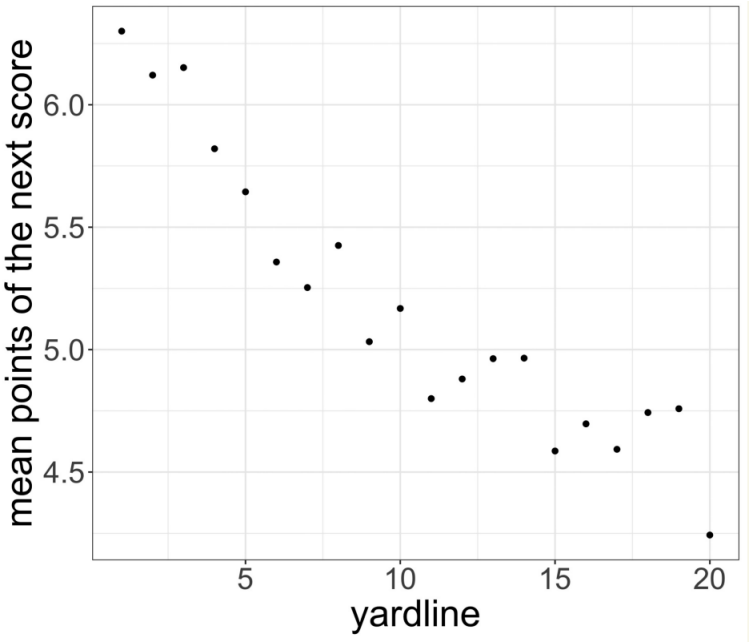

Figure 2.4: Relationship between yard line and net points

As before, we solve for $\widehat{\boldsymbol{\beta}}$ with the normal equations from Equation 2.5:

$$\widehat{\boldsymbol{\beta}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y} \tag{2.12}$$

We turn to `R` to fit our linear and quadratic models.

```
# linear model
l_model = lm(next_score ~ yds_to_end, data = nfl_data)
l_coef = l_model$coefficients
# quadratic model
q_model = lm(next_score ~ yds_to_end + I(yds_to_end^2), data = nfl_data)
q_coef = q_model$coefficients
```

We plot the models in Figure 2.6, and the quadratic model clearly fits the data better.

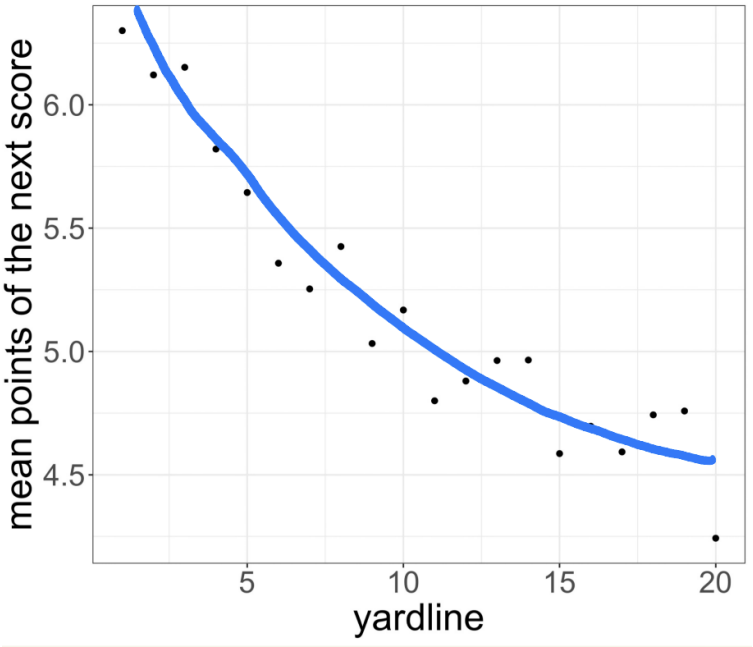

Figure 2.5: Relationship between yard line and net points

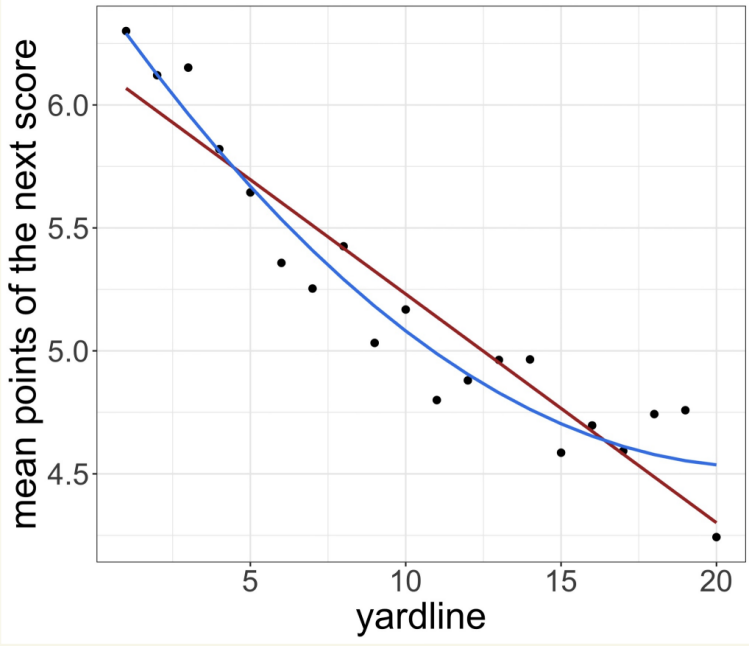

Figure 2.6: Linear and quadratic models fit to the data

2.7 Example: NFL Draft Expected Value Curve

We have a dataset of NFL draft picks. As pictured in Figure 2.7, each row represents a draft pick and contains the following information:

- i: index of the i^{th} draft pick
- X_i: player i's draft pick number
- y_i: player i's first contract "performance value" (Massey-Thaler, 2013)

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

Figure 2.7: Head of the NFL draft dataset

We model the outcome of a draft pick y_i as a function of draft position X_i. Mathematically, we have

$$y_i = f(X_i) + \epsilon_i \tag{2.13}$$

where f is some unknown function and ϵ_i is a mean-zero noise term. As in Section 2.5, we want to estimate the expected value of this quantity:

$$\mathbb{E}[y_i \mid X_i] = f(X_i) \tag{2.14}$$

We proceed by plotting the relationship between draft position and first contract performance value in Figure 2.8.

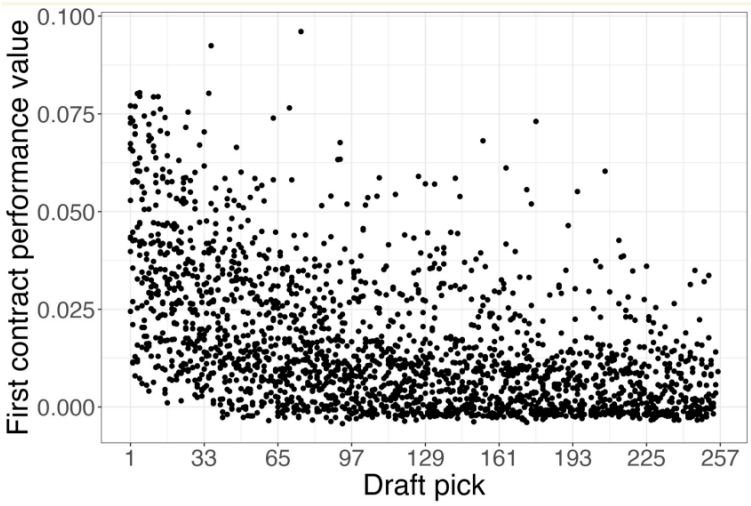

Figure 2.8: Relationship between draft position and first contract performance value

Once again, the expected value $\mathbb{E}(y|X)$ looks nonlinear. In particular, the relationship is convex: the dropoff between picks t and $t+1$ decreases as t increases. How do we select a function to model this?

2.7.1 Splines

To model a general nonlinear relationship, we can use a piecewise polynomial function, or **spline**. To implement this, we fit a separate (usually-cubic) polynomial to different subsections of the data. For example, we could fit a separate cubic polynomial in each round of the draft. Since there are 32 picks per round, our 7 separators (or **knots**) would be at 33, 65, 97, 129, 161, 193, and 225.

To force our fitted spline to be **smooth**, we enforce second-order (or C2) continuity, which means that at each knot, the spline, its first derivative, and its second derivative are all continuous.

2.7.2 Deriving Solution to a One-Knot Spline

Suppose we fit a cubic spline with one knot at $x = k$ (e.g. $x = 129$, at the middle of the draft). We model $y_i = f(x_i|\boldsymbol{\beta}) + \epsilon_i$, where f is the spline and $\boldsymbol{\beta}$ are the spline parameters.

We express the function as a piecewise polynomial:

$$f(x|\boldsymbol{\beta}) = \begin{cases} \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3, & \text{if } x \le k \\ \beta_4 + \beta_5 x + \beta_6 x^2 + \beta_7 x^3, & \text{if } x \ge k \end{cases} \tag{2.15}$$

And we enforce the following continuity constraints:

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

From these constraints, we have the following system of equations:

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

Solving for β_5, β_6, and β_7, we have

$$\begin{aligned} \beta_5 &= (\beta_0 + \beta_1 k + \beta_2 k^2 + \beta_3 k^3 - \beta_4 - \beta_6 k^2 - \beta_7 k^3)/k \\ \beta_6 &= (\beta_1 + 2\beta_2 k + 3\beta_3 k^2 = \beta_5 - 3\beta_7 k^2)/2k \\ \beta_7 &= (\beta_2 + 3\beta_3 k - \beta_6)/3k \end{aligned}$$

And we see that β_5, β_6, and β_7 are completely determined by $\beta_0, \beta_1, \beta_2, \beta_3$, and β_4, so we only need to estimate 5 paramters! These are estimated by forming the model matrix and performing a multivariable regression.

2.7.3 Fitting the Full Spline

Now we fit the full 7-knot cubic spline on the NFL draft data. We use `R` to fit the model.

```
# fit the full spline
full_spline = lm(contract_value ~ splines::bs(draft_pos, degree = 3,
          knots = seq(33, 225, by = 32)), data = draft_data)
# get coefficients
full_coef = full_spline$coefficients
```

However, when we plot the model in Figure 2.9, we notice that our model is a big wiggly at the end of the draft. This indicates that our model is overfitting the data (too many knots, not enough data). To fix this, we can reduce the number of knots to get a smoother fit.

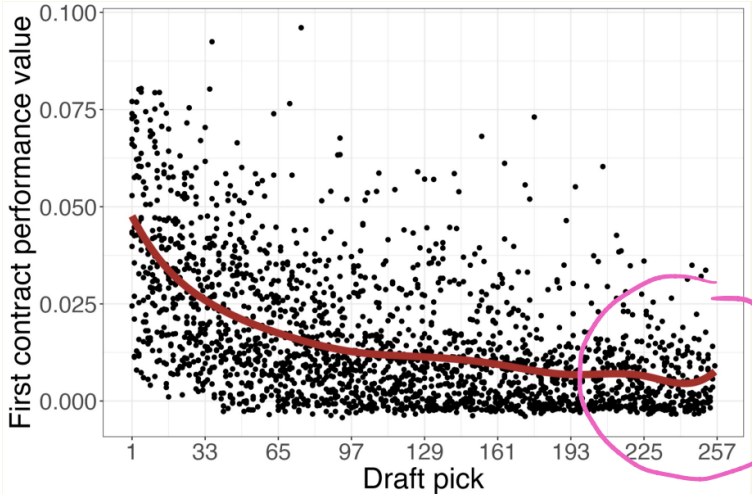

Figure 2.9: Full cubic spline fit to the data.

When fitting splines, the degrees of freedom is the number of parameters we need to estimate. Generally, this formula is

$$\text{df} = \text{number of knots} + \text{degree of polynomial} + 1 \tag{2.16}$$

We can specify the degrees of freedom in `R` with the `df` argument, which along with the `degree` argument, specifies the number of knots. Let's try re-fitting the model with 5 degrees of freedom (3rd degree polynomial with 1 knot).

```
# fit reduced spline with 5 df
red_spline = lm(contract_value ~ splines::bs(draft_pos, degree = 3,
          df = 5), data = draft_data)
# get coefficients
red_coef = red_spline$coefficients
```

When we plot this new spline in Figure 2.10, it's clear we have a much smoother fit that does not overfit to the data.

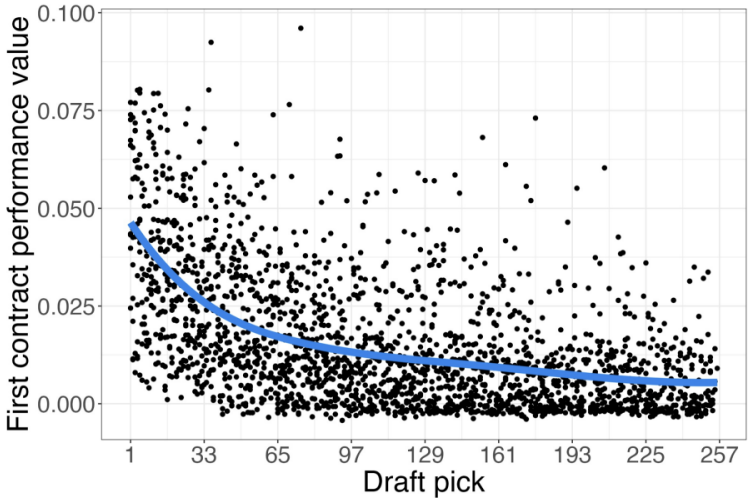

Figure 2.10: Reduced cubic spline fit to the data.

References

[M-T] Massey, C. & Thaler, R., *The Loser's Curse*, Management Science, 2013.