Tennis Logistic

Caleb Skinner

2023-07-28

Contents

Overview

Logistic regression is a statistical model used to explain and analyze the relationship between one dependent (or response) variable and one or more independent (or predictor) variables. It follows many of the guidelines and procedures of simple linear regression *(link to other module).* However, unlike some other regression models, logistic regression's dependent variable must be a categorical variable. The independent variables can be any type of variable. Logistic regression uses the independent variables to predict the outcome of the dependent variable. Given several inputs, it assigns the probability of an outcome ranging from 0 to 1.

For example, a statistics teacher may be interested in predicting which of her students misses class on the day of a test. This is a binary categorical variable: a student either attends or is absent. The teacher can then look compile a list of predictor variables. This could include past attendance rates, homework grades, student GPA, day of the week of the test, illness history, etc. If she keeps track of this data, these variables will help her to predict the students that will miss class. Other logistic regression techniques will help her to identify which variables are important and the level of confidence in her predictions.

Logistic regression is commonly used to predict future outcomes and classify observations. It has numerous practical examples. It is widely used in the medical community to forecast the onset of diseases, in political science to predict elections, in marketing to understand consumer behavior, in banking to catch fraud, and in sports to predict victors.

Tennis Overview

In this module, we'll analyze a sport that is full of dependent variables prime for logistic regression. Tennis's structure is full of binary scoring. For each point, game, set, and match, one player wins and one player loses. There is no third or partial option.

Tennis's scoring system can seem a bit complex for those unfamiliar. In short, two players play one **match**. In mens' tennis, one player must win three **sets** to win a match. In womens' tennis, one player must win two sets. The first player to six **games** (win by two) wins a set, and the first player to four **points** (win by two) wins a game.

One player **serves** to start the point (and the other **returns**) for an entire game, alternating serves each game with their opponent. Players have an advantage when they serve, and it is very significant when a player wins a game that their opponent serves. Winning a game as a returner is called a **break**. Conversly, winning a game as a server is called a **hold**.

Each week, there are single elimination tournaments for these players to compete in. We'll look at tennis's most famous tournament, Wimbledon. Wimbledon begins with 128 players in the men's and women's singles competitions. If the players win, they move on to the next round. After seven rounds, a champion is crowned.

This should be a thorough enough summary for you to understand the contents of the module. If you are still struggling to understand the rules of tennis, please see our data dictionary on the next page for clarity.

Data

Our data includes both mens' and womens' singles tennis matches at Wimbledon in July, 2022. We have data from each point. A full data dictionary explaining each of our variables is below. The score of the match is recorded *before* the point takes place.

variables	explanation	example
match_id	match identification; 1st digit = sex of match; 2nd digit = round	1101, 2701, etc.
player1	first and last name of the first player	Novak Djokovic, Elena Rybakina, etc.
player2	first and last name of the second player	Soonwoo Kwon, Ons Jabeur, etc.
rank_diff	difference in worldwide rankings of player 1 and player 2	78, -21, etc.
match_victor	winner of the match	1 if player 1 wins, 0 if player 2 wins
p1_sets	sets won by player 1	0, 1, or 2
p2_sets	sets won by player 2	0, 1, or 2
p1_games	games won by player 1 in current set	0, 6, etc.
p2_games	games won by player 2 in current set	0, 6, etc.
p1_score	player 1's score within current game	0 (love), 15, 30, 40, AD (advantage)
p2_score	player 2's score within current game	0 (love), 15, 30, 40, AD (advantage)
point_victor	winner of the point	1 if player 1 wins, 0 if player 2 wins
server	server of the point	1 if player 1 serves, 0 if player 2 serves
pt_win_perc	percentage of points won by player 1 in match	47, 51, etc. (first 3 games are set to 50)
roll_win_perc	percentage of last 25 points won by player 1 in match	48, 52, etc. (first 9 points are set to 50)
set_diff	difference in sets won by player 1 and player 2	-2, -1, 0, 1, or 2
p1_break_pt	player 1 has a chance to "break" or win player 2's service game	0 or 1
p2_break_pt	player 2 has a chance to "break" or win player 1's service game	0 or 1
status	measurement of score in set	"no break" if players are even, "p1_break" if player 1 has advantage, "p2_break" if player 2 has advantage
last_point_victor	winner of previous point	1 if player 1 won previous point, 0 if player 2 won previous point
lastpt_p1_winner	player 1 hit an untouchable shot to win the previous point	0 or 1
lastpt_p2_winner	player 2 hit an untouchable shot to win the previous point	0 or 1
p1_serving_for_set	player 1 is serving and has chance to win set	0 or 1
p2_serving_for_set	player 2 is serving and has chance to win set	0 or 1

variables	explanation	example
p1_serving_to_stay	player 1 is serving and could lose set	0 or 1
p2_serving_to_stay	player 2 is serving and could lose set	0 or 1

And here is an example slice of our data. *In ISLE, this will be the full data, allowing them to scroll at their leisure*

match_id	player1	player2	rank_diff	match_victor	p1_sets	p2_sets	p1_games	p2_games	p1_score	p2_score	point_victor	server
1101	Novak Djokovic	Soonwoo Kwon	78	1	0	0	0	0	0	0	0	1
1101	Novak Djokovic	Soonwoo Kwon	78	1	0	0	0	0	0	15	1	1
1101	Novak Djokovic	Soonwoo Kwon	78	1	0	0	0	0	15	15	1	1
1101	Novak Djokovic	Soonwoo Kwon	78	1	0	0	0	0	30	15	1	1
1101	Novak Djokovic	Soonwoo Kwon	78	1	0	0	0	0	40	15	0	1
1101	Novak Djokovic	Soonwoo Kwon	78	1	0	0	0	0	40	30	0	1
1101	Novak Djokovic	Soonwoo Kwon	78	1	0	0	0	0	40	40	1	1
1101	Novak Djokovic	Soonwoo Kwon	78	1	0	0	0	0	AD	40	1	1
1101	Novak Djokovic	Soonwoo Kwon	78	1	0	0	1	0	0	0	0	0
1101	Novak Djokovic	Soonwoo Kwon	78	1	0	0	1	0	0	15	1	0
1101	Novak Djokovic	Soonwoo Kwon	78	1	0	0	1	0	15	15	1	0
1101	Novak Djokovic	Soonwoo Kwon	78	1	0	0	1	0	30	15	0	0
1101	Novak Djokovic	Soonwoo Kwon	78	1	0	0	1	0	30	30	0	0
1101	Novak Djokovic	Soonwoo Kwon	78	1	0	0	1	0	30	40	1	0
1101	Novak Djokovic	Soonwoo Kwon	78	1	0	0	1	0	40	40	0	0
1101	Novak Djokovic	Soonwoo Kwon	78	1	0	0	1	0	40	AD	0	0
1101	Novak Djokovic	Soonwoo Kwon	78	1	0	0	1	1	0	0	1	1
1101	Novak Djokovic	Soonwoo Kwon	78	1	0	0	1	1	15	0	0	1
1101	Novak Djokovic	Soonwoo Kwon	78	1	0	0	1	1	15	15	0	1
1101	Novak Djokovic	Soonwoo Kwon	78	1	0	0	1	1	15	30	0	1
1101	Novak Djokovic	Soonwoo Kwon	78	1	0	0	1	1	15	40	0	1

Univariate Logistic Regression

Let's begin by using one predictor variable to predict one response variable.

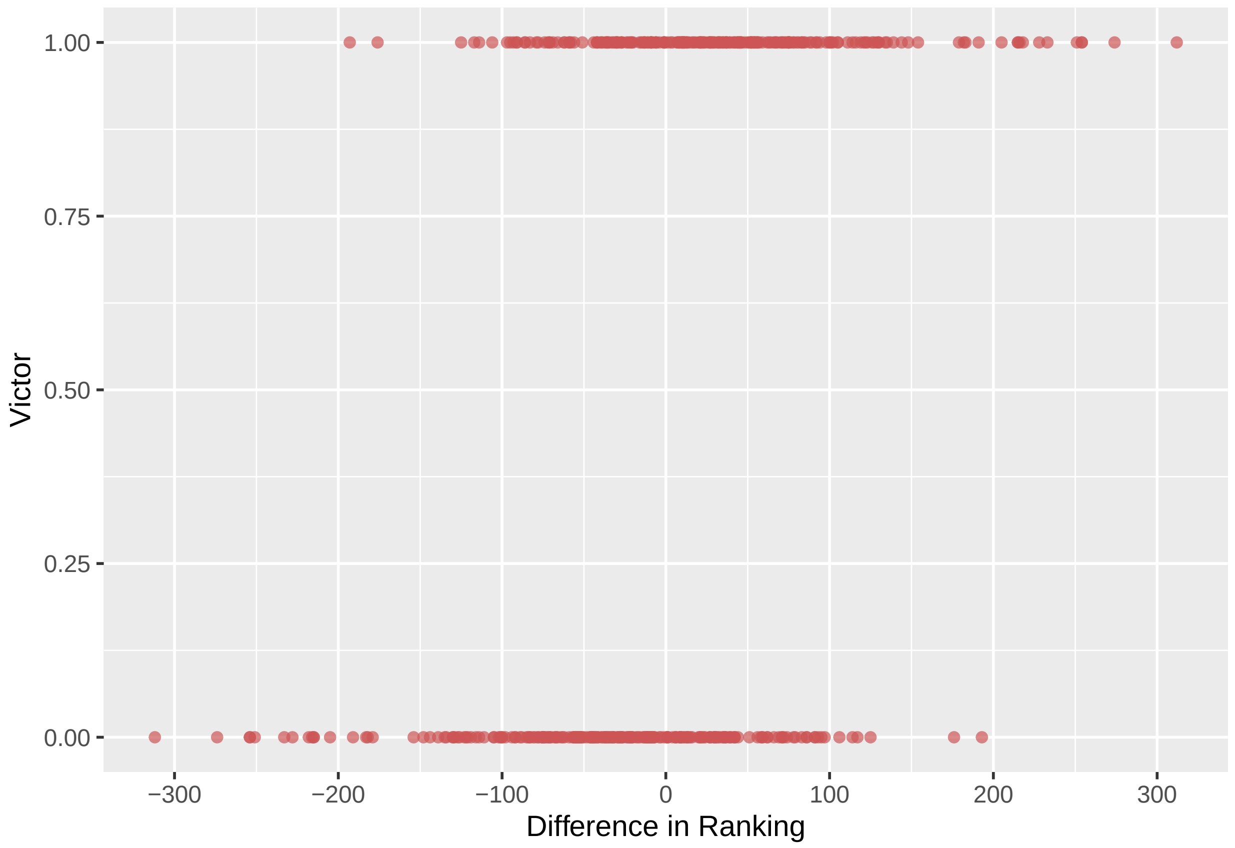
Binary Response Variables

The identifying factor of logistic regression is the binary response variable. A **binary response variable** is a dependent variable that only has two outcomes. Correlation with binary variables can be difficult to plot, understand, and interpret.

Let's begin by reducing the data to one observation for each match. We can now assess the binary response variable match_victor. The result is 1 if the player wins and 0 if the player loses.

Let's try to predict the result of match_victor by using the discrete predictor variable rank_diff. Rank_diff is the difference between the two players' worldwide rankings. Below is a scatter-plot that visualizes their relationship. The players with superior rankings are located on the right-hand side, and the players with inferior rankings are on the left.

I duplicated each match, by making each player an observation. Should I not have duplicated? This helps things by making the intercept 0 and making it more perfectly symmetrical. I didn't duplicate the later data sets



I am kinda uncertain whether to call the variable Victor vs match_victor. I feel like Victor is easier to understand, but it doesn't follow the same format as the other variables I use in the rest of the module. I just impulse changed it to match_victor

The stack of points where match_victor (V) = 0 represent the percentage of points won by losing players at Wimbledon. Conversely, the stack of points where match_victor (V) = 1 represent the percentage of points won by victorious players at Wimbledon.

Would you estimate a player ranked 100 spots below his or her opponent wins his/her match? How confident would you be?

What about 50 spots below his or her opponent? 200? 25?

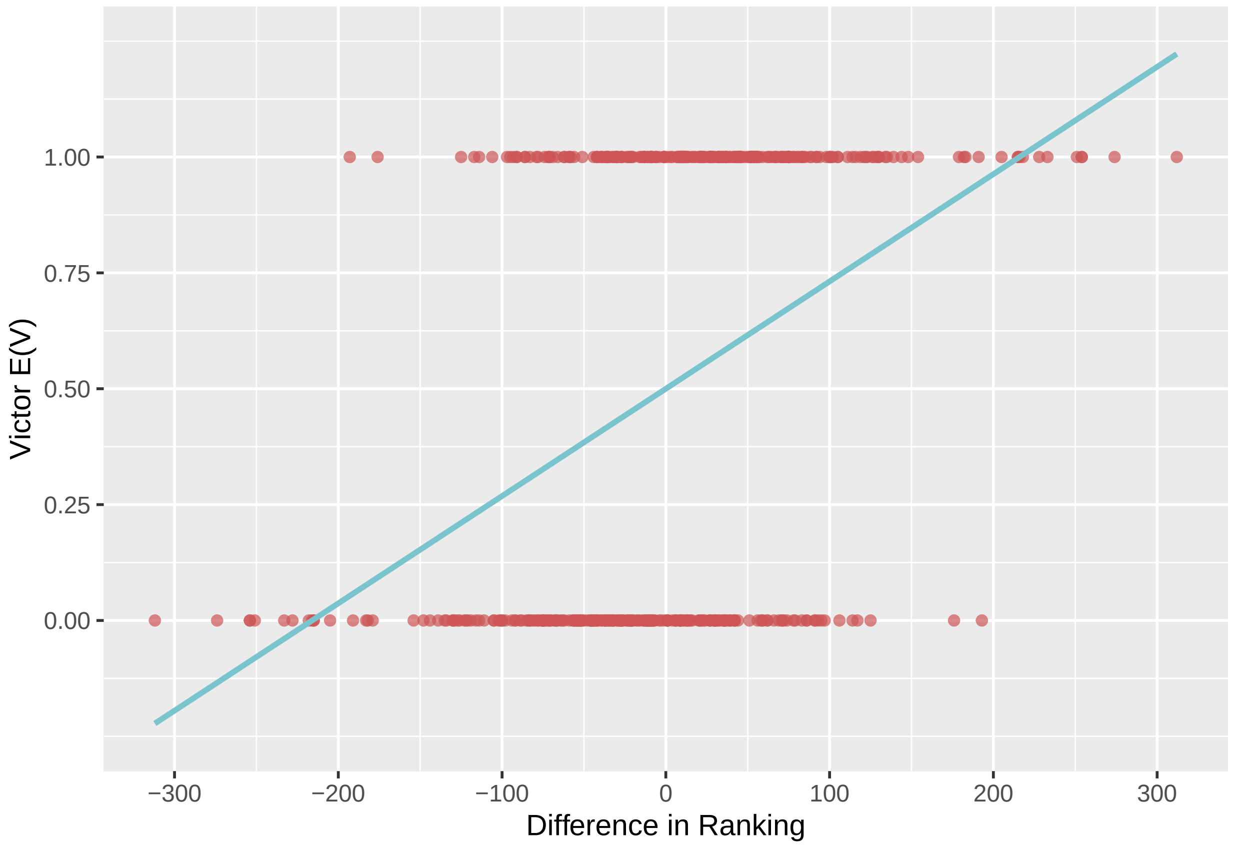
In simple linear regression, we place a linear trend line on the scatter plot so that our squared residuals are minimized. The **simple linear regression function** follows the formula:

- $E(Y) = \beta_0 + \beta_1 X_1$

where β_0 and β_1 are parameters and E(Y) is the **expected value** of Y, our response variable. Expected value is an important concept in regression with binary response variables. It represents the average value of the binary response variables given the predictors. These expected values will manifest in proportions ranging from 0 to 1.

For example, if, given a set of predictors, 8 out of 50 players win the match, then the expected value of match_victor (V) is $\frac{8}{50} = 0.16$. On average 16% of players win the match. In other words, there is a 16% probability that V = 1 given the set of predictors.

Let's see what a linear regression model looks like on our data. Remember, the linear trend line represents the expected value of match_victor for our data.



Where does the E(V) approach 0? Where does it approach 1?

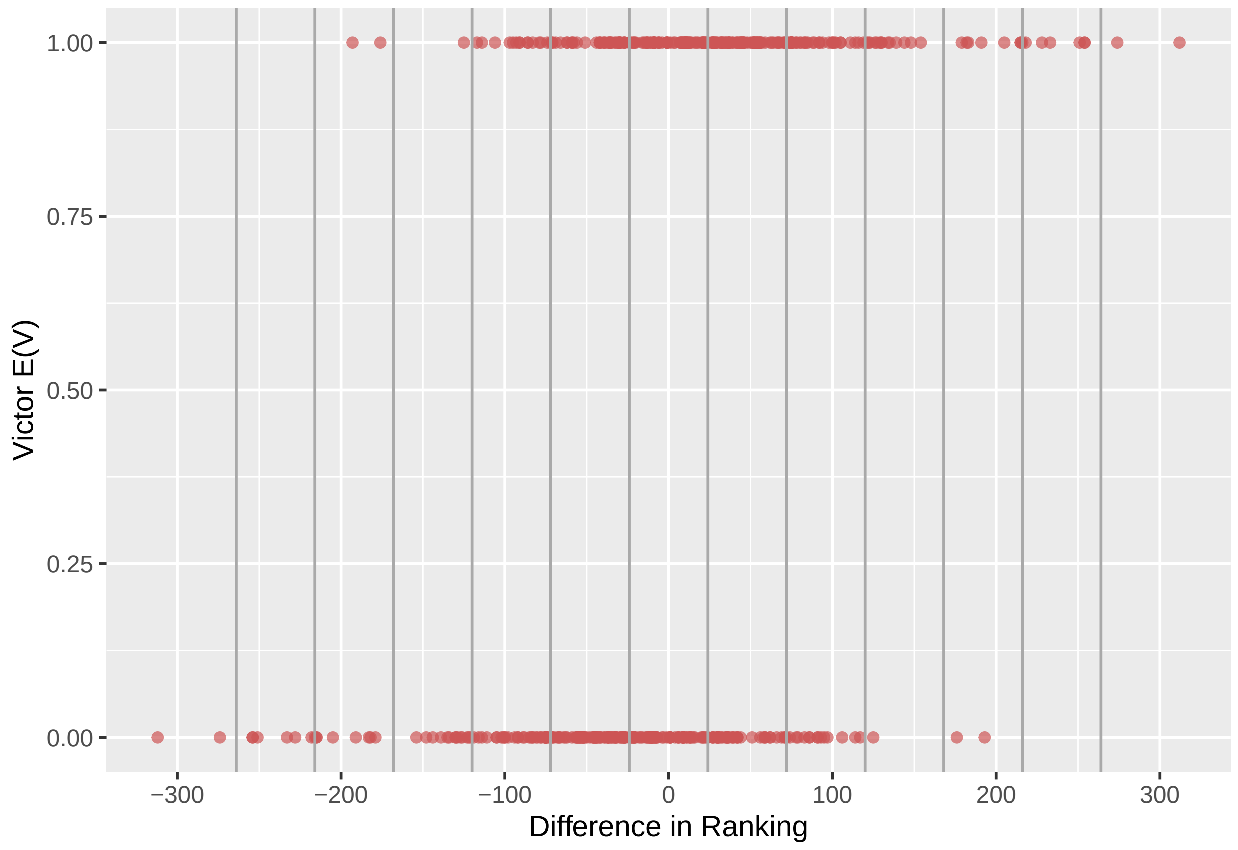
What is the E(V) when the player is ranked 100 spots below his or her opponent? Is this close to what you estimated? What about 200 spots?

What is the E(V) when the player is ranked 300 spots below his or her opponent? Is this problematic?

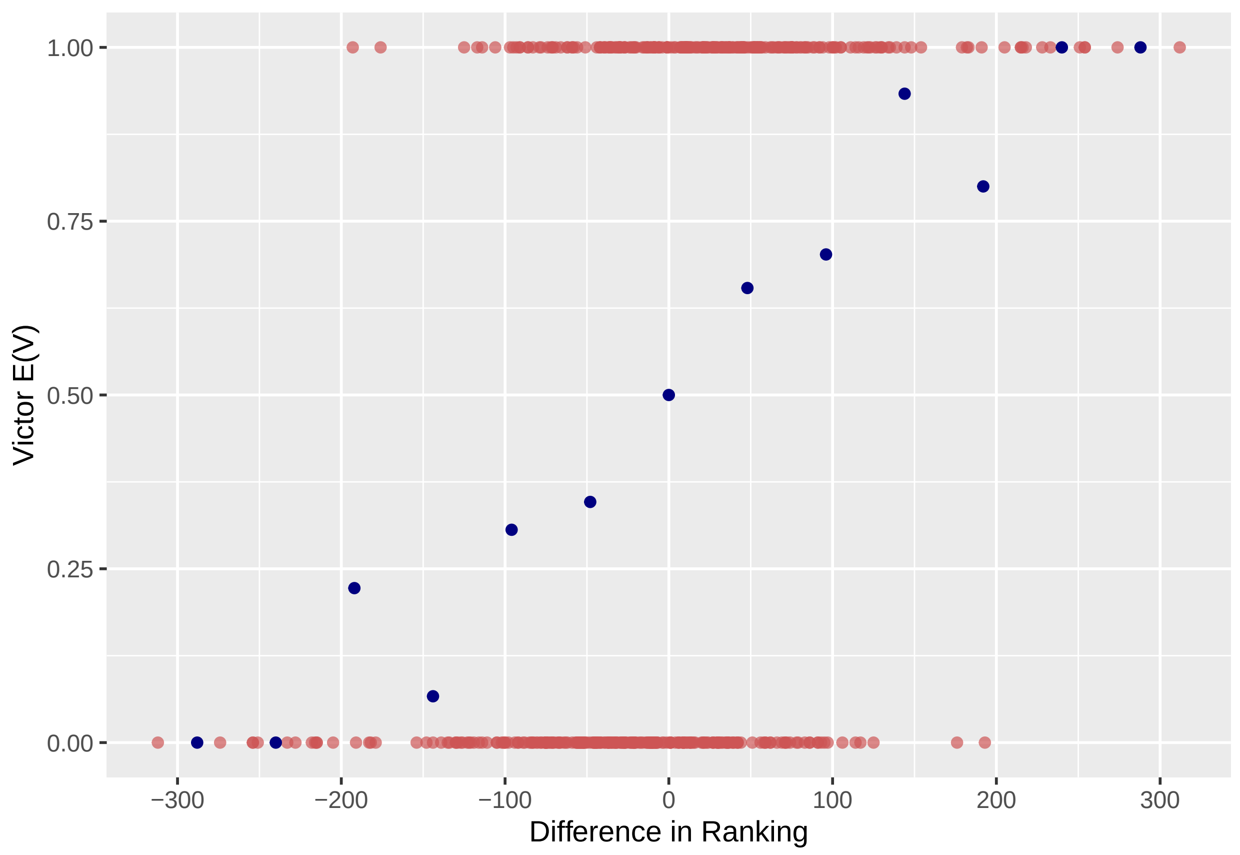
How would you assess the linear regression model as a whole? Do the expected values match your expectations?

Bins

The linear model is not a good fit for our data. It underestimates E(V) on both ends of the model. One way to improve the model is to separate the data into several groups called **bins**. We displayed the same visualization as before, but we added thirteen bins to help separate up the data.



We can find the proportion of victories within each bin and plot them. The dots in blue are the proportion of victories within each bin.



The proportions of each bin form a curve. The shape of this curve is typically "S" shaped. The change in the proportion per unit decreases as the E(V) approaches 0 or 1. This is because the victory probability can never surpass the 0 to 1 range. As the players' ranking increases or decreases, eventually the E(V) must flatten.

Logistic Visualization

There are several methods developed to model this "S" curve, but logistic regression is a popular choice, because it is mathematically easy to manipulate and it has meaningful and simple interpretations.

Logistic regression follows the logistic function:

- $E(Y) = \pi(x) = \dfrac{e^{\beta_0+\beta_1 x}}{1+e^{\beta_0+\beta_1 x}}$

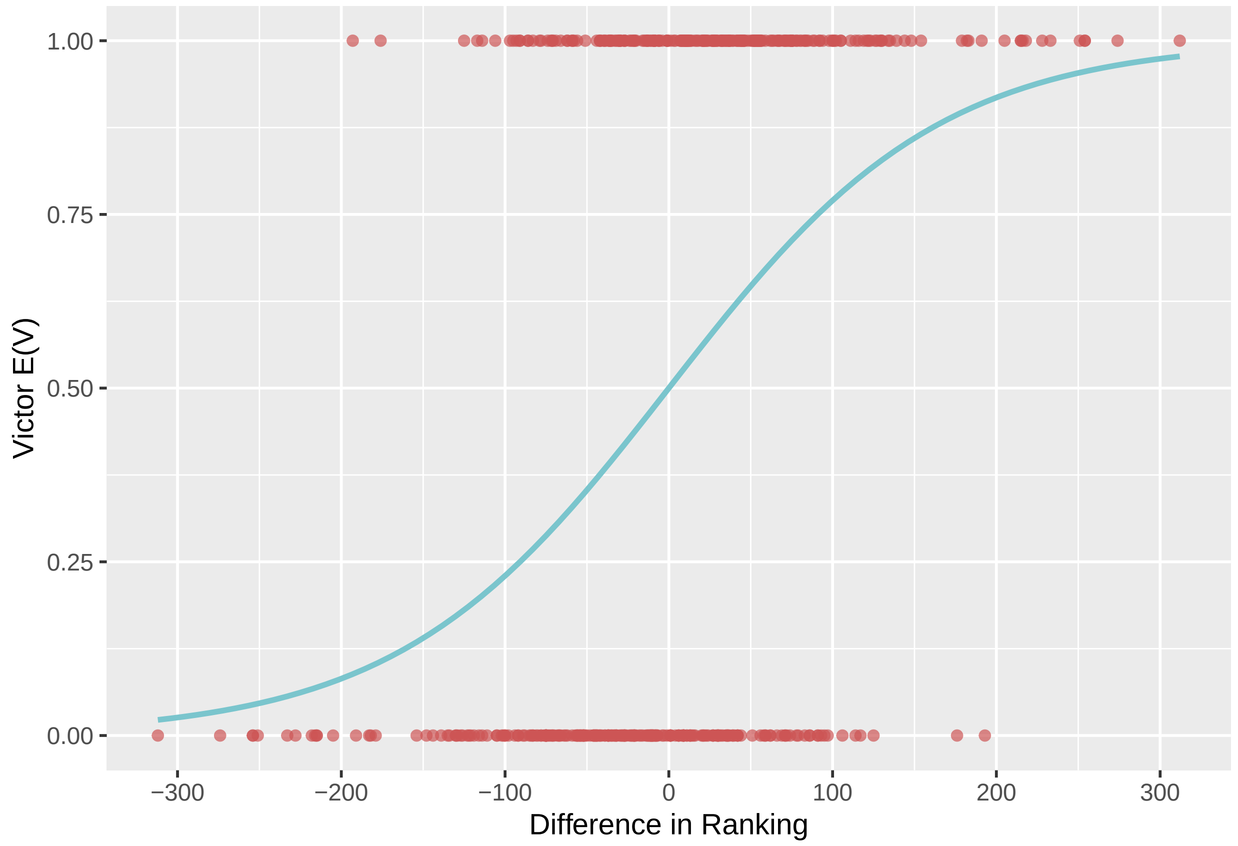
where $\pi(x)$ is the probability of y given x.

keep below?

The **likelihood** is the probability of obtaining the observed set of data given the parameter estimates. Logistic regression uses an estimation method called **maximum likelihood**. It creates estimators $\hat{\beta_0}$ and $\hat{\beta_1}$ for the unknown true parameters β_0 and β_1 in order to maximize the likelihood.

It is common for logistic regression models to use the **log likelihood** (log of the likelihood), because it is easier to work with. This value will always be negative, and the closer the log likelihood is to 0, the more closely the estimators fit on the data.

Let's see what a logistic regression would looks like mapped onto our data. The blue line represents the expected value of match_victor. It's our models predicted victory probability.



Assess the logistic model. Can you see the "S" shaped curve? How well does it match the proportions? How does it differ from the linear model?

What is the proportion of victories for a player ranked 100 spots below his or her opponent? What about 50 spots below? 200?

Predictions and Residuals

Our logistic model also provides us with estimates for our parameters β_0 and β_1.

- $\hat{\beta}_0$ (Intercept): 0
- $\hat{\beta}_1$ (rank_diff): 0.01014

Thus,

$$\hat{\pi}(x) = \frac{e^{0+.01014x}}{1+e^{0+.01014x}}$$

With our logistic function, we can predict a value for V given any value of x. Try a few below:

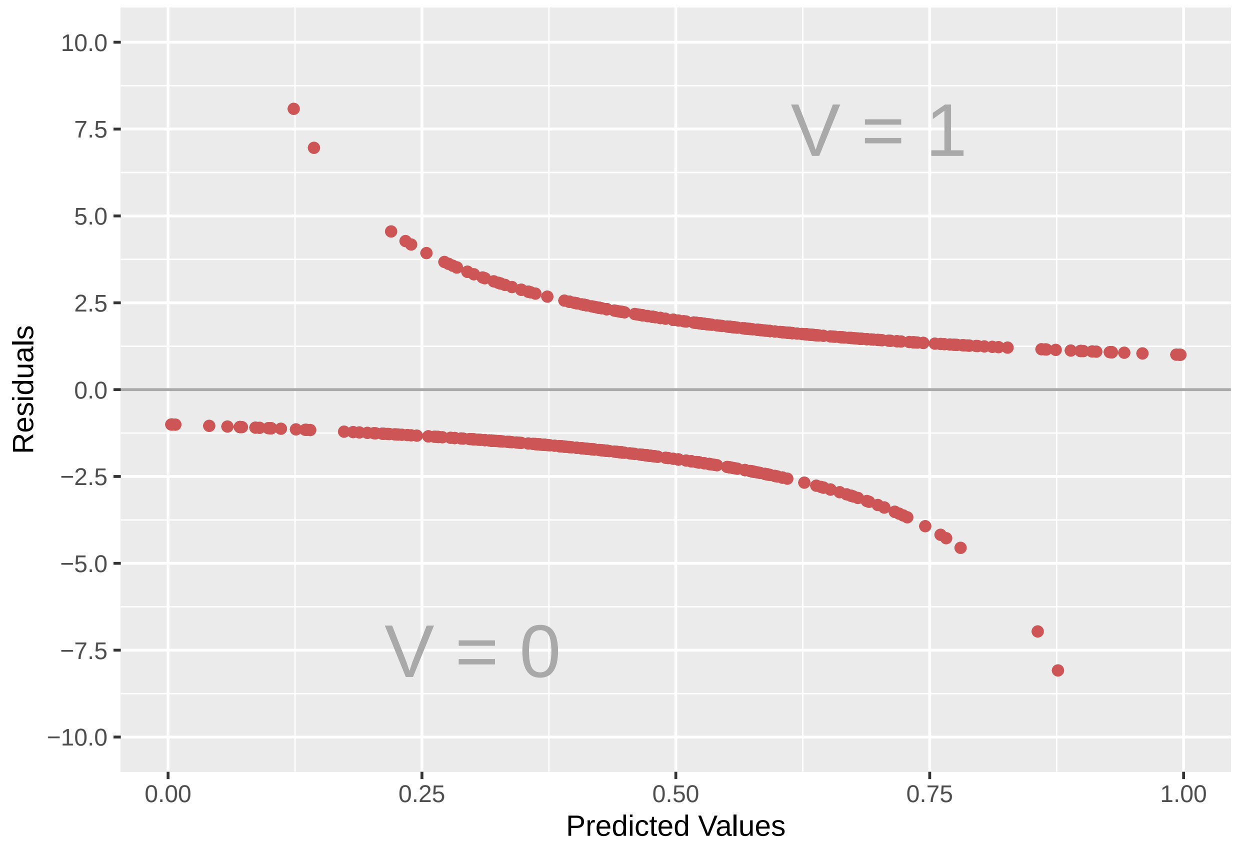
student can enter a value of x and it should spit out a $\pi(x)$. I started with 50

$\hat{\pi}(x)$: 0.624

However, like all models, the logistic function cannot perfectly predict the data. We can add the error term ϵ to our function to represent the difference between our estimation $\hat{\pi}(x)$ and the binary result V:

- V $= \hat{\pi}(x) + \epsilon$

A **residual** is the difference between the observed value (V) and the model's expected or predicted value ($\hat{\pi}(x)$). We can plot these values on a residual plot. The residuals are on the "logit" scale. We'll come back to that later.



Each residual represents the error term $\hat{\epsilon}$. As the difference between the predicted values and actual values increase, the residuals increase. The values furthest from 0 represent the most unlikely events according to our model. Can you guess where those values reside on our scatter-plot?

Interpretation

Interpreting the logistic model relies heavily on the understanding of odds. **Odds** are a ratio of the probability of a success and the probability of a failure (1 - success). We will often need to use this conversion in our interpretations. In our case:

- $odds(x) = \dfrac{\pi(x)}{1 - \pi(x)}$

which conveniently simplifies to:

- $odds(x) = e^{\beta_0 + \beta_1 x}$

Often, we want to find how these odds increase or decrease as our predictor changes. This is called the **odds ratio (OR)**. The odds ratio is calculated by the proportion of odds(x)/odds(x + 1).

- OR $<$ 1 – an increase in x decreases the odds of Y = 1.
- OR = 1 – an increase in x does not impact the odds of Y = 1.
- OR $>$ 1 – an increase in x increases the odds of Y = 1.

Recall our $\hat{\beta_0}$ and $\hat{\beta_1}$. Can you find the player's odds of victory where x = 50? x = 51? Can you calculate their odds ratio?

I made a function to calculate the odds ratio, should I let the student use it?

Does the odds ratio change when x = 99 and x = 100?

Our model is logistic. This means that for each value of x, the odds ratio remains constant. We can calculate the **log odds** by applying a natural log (ln) to our odds function. This returns:

- log odds or "logit" = $g(x) = \beta_0 + \beta_1 x$

This logit function is strikingly similar to the linear function, and it offers simple, linear-esque- interpretations. β_0 and β_1 are the intercept and slope of the log odds.

In our example,

- $\hat{\beta_0} = 0$ is the estimated log-odds of victory when the difference in ranking is 0.

We can quickly solve for the odds by adding "e" to both sides.

- $e^{\hat{\beta_0}} = e^0 = 1$ is the estimated odds of victory when the difference in ranking is 0. After converting our odds into a probability, we find that $\hat{\pi}(x) = 0.50$.

Often, the intercept is meaningless or out of the natural range of the predictor. Consider a predictor like serve speed, height, or weight; these would never be zero. However, the slope is a crucial piece of analysis.

In our example,

- $\hat{\beta_1} = 0.01014$. For each one unit increase in the difference in ranking, the log-odds of victory is estimated to increase by 0.01014.

Once again, we can provide helpful interpretations by adding an "e" to both sides.

- $e^{\hat{\beta_1}} = e^{.01014} = 1.01019$. The estimated odds of victory increase by 1.02% for each unit increase in ranking difference.

Multivariate Logistic Regression

is this going to cause an issue with independence?

question on conditional effect in regression

The logistic regression model is most useful when applying several predictors to a response variable. This can increase the model's predicting power and reduce the residuals. It can also allow the user to compare the strength of several predictors.

The multivariate logistic regression is adjusted to allow for more predictors:

- $E(Y) = \pi(x_1, x_2, ...x_p) + \epsilon = \dfrac{e^{\beta_0+\beta_1 x_1+...+\beta_p x_p}}{1+e^{\beta_0+\beta_1 x_1+...+\beta_p x_p}} + \epsilon$

where $\pi(x_1, x_2, ..., x_p)$ is the probability of y given the set of predictors x.

Once again, the logistic regression uses maximum likelihood estimates. Each of our unknown parameters $\hat{\beta_0}$, $\hat{\beta_1}, \ldots$, $\hat{\beta_p}$ are estimated in order to maximize the probability of obtaining the observed set of data.

Earlier, we used the difference in ranking predictor to assess the match_victor binary response variable. Now, we will use a myriad of predictors to assess the point_victor binary response variable.

In tennis, there are hundreds of points played in a match. Often the difference between winning and losing can be only a few points. Over the course of Wimbledon in 2022, the losing player still won an average of 45.63% of his or her points. But the circumstances of each point can be very different. Certain scores create more tense situations, and, often, the results of previous points can linger in players' minds. Players also typically enjoy a distinct advantage when they serve.

For this particular model, we are estimating the winner of a point, and we will use three predictor variables: x_1, x_2, and x_3.

- point_victor (Y): point_victor = 1 if player 1 wins the point, 0 if player 2 wins
- server (x_1): server = 1 if player 1 serves, 0 if player 2 serves
- pt_win_perc (x_2): percentage of the points won in the match at the start of the point (each unit is 1%)
- rank_diff (x_3): difference in the two players' worldwide rankings (player 2 - player 1)

It's important to note that our model is from the perspective of player 1. All point victory probabilities will be from his or her perspective. Solving for player 2's point probabilities will be the complement (1 - $\pi(x)$).

Let's see what the model produces and practice our interpretations. Our model yields the following estimators for our three parameters. They are interpreted in the same manner as in the univariate model.

(Intercept)	rank_diff	server	pt_win_perc
-1.3312	0.0007	1.0337	0.0163

The coefficients, $\hat{\beta_j}$, are the change in log-odds of victory (for player 1) given a one unit increase in the variable.

For instance, if the difference in ranking increases by 1 unit, then the estimated log-odds of victory increases by 0.0007. As with before, we can convert this to odds with e. In this case, for each unit increase in rank_diff, the estimated odds of victory increases by 1.0007.

do I capitalize server here?

Server is an **indicator** variable. Indicator variables are binary; there are only two outcomes. This can make the interpretation of indicator variables seem a little different, because a 1-unit change completely flips the outcomes. Thus, for server, the coefficient directly compares the estimated log-odds of victory when player 1 serves to the estimated log-odds of victory when player 2 serves

Using the previous examples, interpret server and pt_win_perc. What are the changes in log-odds? Changes in odds? Are they meaningful?

The intercept is the log-odds of victory when all values are 0. What is this value? Is it meaningful?

*maybe add a note in isle: pt_win_perc is in percentage form. This means a 1 unit increase is actually 1%.

Goodness of Fit

The estimators of our model give us important pieces of information about the relationship between our binary response variable and our predictors. They cannot, however, evaluate how well the model fits our data.

Goodness of fit assesses the extent in which our model's predicted values match the observed values. We can begin to compute a model's fit by measuring the model's residuals. A model **overestimates** the data if the observed values fall short of the expected values (producing a negative residual). A model **underestimates** the data if the observed values exceed the expected values (producing a positive residual).

Let's say our model gives a player a 70% chance to win a point. If he wins, the model has underestimated his chances and the observation returns a positive residual. If he loses, the model has overestimated his chances and the observation returns a negative residual.

Overall, we want these residuals to be as close to zero as possible. We also, however, want the model to be consistent in its fitness. We don't want a model that systematically overestimates certain types of observations and underestimates others. These systematic problems with the model would suggest a weakness in the model.

The Hosmer-Lemeshow goodness of fit test sorts the observations into ten equally sized groups or **deciles** by their predicted probabilities. Each decile will have similar values of $\hat{\pi}(x)$. The test compares the total expected values of each decile to the total observed values of each decile. Naturally, some of the deciles will be overestimated and some underestimated.

Here are the deciles of our point prediction model. The observed and expected are the number of points won in that category.

Table 4: Point Prediction Deciles

Predictions Range	Observed	Expected	Residual
[0.243,0.352]	1,499	1,522.9	-23.9
(0.352,0.368]	1,657	1,655.8	1.2
(0.368,0.379]	1,655	1,717.6	-62.6
(0.379,0.396]	1,811	1,767.1	43.9
(0.396,0.493]	1,957	1,915.8	41.2
(0.493,0.601]	2,679	2,653.9	25.1
(0.601,0.619]	2,735	2,798.3	-63.3
(0.619,0.629]	2,888	2,860.5	27.5
(0.629,0.645]	2,897	2,917.4	-20.4
(0.645,0.783]	3,078	3,046.6	31.4

Notice that some of the deciles are overestimated and some are underestimated by the model.

The Hosmer-Lemeshow test evaluates these residuals and determines if they are large enough to suggest there are problems with our model. Our hypotheses are structured differently than traditional tests:

- H_0: is that the model fits the data.
- H_a: is that the model does not fit.

Our test statistic, X^2, can be solved with the following formula:

$X^2 = \sum \frac{(observed - expected)^2}{expected}$
across our ten deciles.

If the deciles contain residuals that are far from zero, then we will have a large X^2. A large X^2 indicates a substantial problem with the model and suggests there are inconsistencies across its predictions. We would reject the null hypothesis that the model fits.

The Hosmer-Lemeshow test for our data:

- X^2: 13.8029759
- p-value: 0.08705

Since the p-value is larger than our alpha α of .05, we fail to reject our null hypothesis that the model fits.

Regardless of the overall fit of the model on the data, it is important to check the data for **influential** observations. Two important types of influential observations are outliers and leverage points. **Outliers** are observations with an extreme residual. They are points that the model struggled to predict. **High leverage** observations typically have uncommon characteristics. Their uniqueness causes them to have a strong impact on the model.

We don't have time to delve into the complexities of these points for now. If you are creating a logistic regression for yourself, it is important to check these observations and learn their characteristics. Sometimes, they hold interesting results or represent inaccurate data.

Likelihood Ratio Test

Thus far, our model contains three predictors, but there are plenty of other variables we could add to improve the model. We also want to keep the model simple and avoid weak predictors. How do we decide when to add a variable and when to refrain?

When making this decision, it's helpful to imagine two distinct models:

1. Our *simple* model lacks the variable(s) in question.
2. Our *complex* model includes the variable(s) in question.

The **likelihood ratio test** compares the log likelihood between the two models. Recall that the log likelihood is a measure of the predictive ability of a model. If the difference between the two models is large enough, we can conclude that the variable(s) in question is significant in predicting the outcome.

In a simplified form, our hypothesis for the likelihood ratio test are as follows:

- H_0: the simple model is preferable.
- H_a: the complex model is preferable.

Let's say that we are considering adding a fourth variable, set_diff, to our model. We can use the likelihood ratio test to assess the difference in log likelihoods of our complex model with set_diff and our simple model without set_diff.

Below is our code and its output. You will need to reference it later.

```r
# producing simple model termed "simple"
simple <- glm(point_victor ~ rank_diff + server + pt_win_perc,
                  family = binomial(link = "logit"), data = prep_w2022)
# producing complex model termed "complex" - adding set_diff to equation
```

```r
complex <- glm(point_victor ~ rank_diff + server + pt_win_perc + set_diff,
                 family = binomial(link = "logit"), data = prep_w2022)
# likelihood ratio test
lrtest(simple, complex)
```

```
## Likelihood ratio test
##
## Model 1: point_victor ~ rank_diff + server + pt_win_perc
## Model 2: point_victor ~ rank_diff + server + pt_win_perc + set_diff
##   #Df LogLik Df  Chisq Pr(>Chisq)
## 1   4 -30254
## 2   5 -30250  1 6.6809   0.009745 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The log likelihood for our simple model is -30254 and for our complex model is -30250. Our p-value is .009745, and this is less than our alpha of .05. Therefore, we should reject the null hypothesis and use the complex model.

We have several other variables that we'd like to check. Some of the variables are pairs, meaning that one applies to player 1 and one applies to player 2. Because there is no inherent difference between player 1 and player 2, it is best to keep the paired variables together in a model.

thoughts? ^

- p1_break_pt and p2_break_pt
- roll_win_perc
- status (categorical variable with three categories - interpretation may be advanced)
- last_point_victor
- lastpt_p1_winner and lastpt_p2_winner
- p1_serving_for_set and p2_serving_for_set
- p1_serving_to_stay and p2_serving_to_stay

I am uncertain about the three "last" variables, because technically they make the data set smaller, because the first point of each match returns NA values. This will mess with likelihood ratio tests and such. I converted the NAs to 0, which is technically inaccurate...

These are all variables that we have identified as potential predictors for our model. If you can't remember their meaning, then please check the data dictionary. Some variables are more complicated than others. Make sure to choose a variable that you can understand.

Using the code chunk below, create a simple and complex model and test the likelihood ratio of both of them. You can replicate our code from above. Copy the code exactly and then add your own variables using the + operator. Make sure the variables are spelled exactly like they are displayed. Repeat this process several times until you think you've found the best model for our data.

here is an example of what the student would do

```
## Likelihood ratio test
##
## Model 1: point_victor ~ rank_diff + server + pt_win_perc + set_diff
## Model 2: point_victor ~ rank_diff + server + pt_win_perc + set_diff +
##     last_point_victor
##   #Df LogLik Df Chisq Pr(>Chisq)
## 1   5 -30250
## 2   6 -30250  1 0.495     0.4817
```

Once you've evaluated the test, interpret it and decide if we have enough evidence to reject our null hypothesis.

The likelihood ratio test can also be used to test the significance of the entire model. It follows a similar to what we have described and is an important and common use. However, we don't have time to cover it in this module.

Testing Individual Parameters

The likelihood ratio test assesses the strength of the entire model, but sometimes we want to focus our attention on one variable.

The **Wald test** assesses the effect of one predictor on the binary response variable. The test "controls" for the influence of each other variable in the model; the effect of all other variables in the model are held constant. If the variable's slope β_j is significantly far from 0, then we can conclude that there is an effect on the response variable.

Our hypotheses for the Wald test:

- $H_0 : \beta_j = 0$ (no effect)
- $H_a : \beta_j \neq 0$ (some effect)

Our test statistic Z is calculated simply with the formula:

- Z $= \dfrac{\hat{\beta}_j}{SE(\hat{\beta}_j)}$

Our statistical software will display the Wald test results for each parameter in a table. Let's look specifically at the pt_win_perc variable in our new model (with set_diff) to see if its effect is significant.

should I display just the one variable or all of them here?

Table 5: Point Parameters Wald test

Parameter	Estimate	Std. Error	z value	p.value
(Intercept)	-1.1858	0.1016	-11.6696	1.8e-31
rank_diff	0.0007	0.0001	6.7915	1.1e-11
server	1.0328	0.0194	53.1896	0
pt_win_perc	0.0134	0.0020	6.7319	1.7e-11
set_diff	0.0333	0.0129	2.5845	0.0098

Wald's test statistic for pt_win_perc is 6.732 and the p-value approaches 0. Our p-value is less than our alpha of .05, so we can reject our null hypothesis and conclude that pt_win_perc has some effect. While controlling for the difference in ranking, the server, and the set score, the proportion of points won in the match has some effect on the player's probability of winning a point.

All four of our predictors have statistically significant effects on point_victor, but this isn't true across every one of our potential variables. If a variable appears to have insignificant effects, we would consider removing it from our model. Before removing a predictor, we should check the likelihood ratio test to confirm our findings.

Try out different combinations of variables for yourself. Which variables are always significant? How does the significance of some predictors vary across different models.

student will be given a coding chunk. This will allow them to test out different variables

Table 6: Point Parameters Wald test

Parameter	Estimate	Std. Error	z value	p.value
(Intercept)	-1.1197	0.1273	-8.7960	1.4e-18
rank_diff	0.0007	0.0001	6.6722	2.5e-11
server	1.0510	0.0230	45.7527	0
pt_win_perc	0.0088	0.0026	3.3719	7e-04
set_diff	0.0392	0.0133	2.9462	0.0032
p1_break_pt	0.0895	0.0483	1.8524	0.064
p2_break_pt	-0.1165	0.0481	-2.4216	0.0155
roll_win_perc	0.0031	0.0014	2.1283	0.0333
statusp1_break	0.0133	0.0307	0.4337	0.6645
statusp2_break	0.0145	0.0305	0.4747	0.635
last_point_victor	-0.0022	0.0241	-0.0925	0.9263
lastpt_p1_winner	0.0017	0.0299	0.0567	0.9548
lastpt_p2_winner	0.0141	0.0293	0.4800	0.6312
p1_serving_for_set	0.0329	0.0600	0.5479	0.5838
p2_serving_for_set	-0.0697	0.0584	-1.1944	0.2323
p1_serving_to_stay	-0.0448	0.0462	-0.9696	0.3322
p2_serving_to_stay	-0.0088	0.0467	-0.1890	0.8501

Interpret at least one new predictor.

The Wald test can occasionally yield different results than the likelihood ratio test, because they use different methodologies. It is generally preferable to defer to the likelihood ratio test's results.

Confidence Intervals

Each of our coefficients $\hat{\beta}_j$ are estimates of the true unknown parameters β_j. The Wald test also provides **confidence intervals** to help quantify the uncertainty of these estimates. Our Wald confidence intervals are calculated with the following formula:

- 95% confidence interval of $\beta_j = \hat{\beta}_j \pm z_{1-\frac{\alpha}{2}} * \hat{SE}(\hat{\beta}_j)$

With 95% confidence, we can assert that our true parameter lies within this interval. These intervals can help us to interpret the magnitude and certainty of our parameters effect. A large confidence interval would indicate that we are uncertain of the parameter's true effect on the response variable.

Let's look to our model. We'll interpret the intercept, pt_win_perc, and server for you, and then ask you to do the same for a few predictor variables. Remember, the coefficients represent the change in log-odds of victory for player 1.

Table 7: Log-Odds Confidence Intervals

Parameter	2.5%	Estimate	97.5%
(Intercept)	-1.3850	-1.1858	-0.9867
rank_diff	0.0005	0.0007	0.0009
server	0.9947	1.0328	1.0708
pt_win_perc	0.0095	0.0134	0.0173
set_diff	0.0080	0.0333	0.0585

Log-odds are difficult to interpret. For more applicable interpretations, we will convert each of our intervals into esimated odds-ratio by applying "e" to each value.

Table 8: Odds-Ratio Confidence Intervals

Parameter	2.5%	Estimate	97.5%
(Intercept)	0.2503	0.3055	0.3728
rank_diff	1.0005	1.0007	1.0009
server	2.7039	2.8089	2.9177
pt_win_perc	1.0095	1.0135	1.0175
set_diff	1.0080	1.0339	1.0602

Intercept: The estimated odds of victory *for player 1* are 0.3055 when all variables are at 0. In this scenario, the two players have the same ranking, player 2 is serving, player 1 has won 0% of the points so far in the match, and the set-score is tied. The odds may be as low as 0.2503 or as high as 0.3728 with 95% confidence.

capitalize these variables?

pt_win_perc: The estimated odds of victory *for player 1* increase by 1.35% for each 1-unit increase in the percentage of points won by player 1 in the match, controlling for the difference in ranking of the players, the server, and the set score. The increase may be as low as 0.95% or as a high as 1.75% with 95% confidence.

Server: The estimated odds of victory *for player 1* increase by 180.89% when player 1 serves, controlling for the difference in ranking of the players, the percentage of points won thus far in the match, and the set score. The increase may be as low as 170.39% or as a high as 191.77% with 95% confidence.

Using our format, interpret a few of the confidence of intervals in your own model. We'll allow you to create your own model using our variables. We'll print out the odds-ratios for each variable. Remember, an odds-ratio above 1 indicates a positive effect on the result variable, and an odds-ratio below 1 indicates a negative effect on the result variable.

guess some of the student models

Table 9: Odds-Ratio Confidence Intervals

Parameter	2.5%	Estimate	97.5%
(Intercept)	0.2543	0.3264	0.4189
rank_diff	1.0005	1.0007	1.0009
server	2.7346	2.8605	2.9925

Table 9: Odds-Ratio Confidence Intervals

Parameter	2.5%	Estimate	97.5%
pt_win_perc	1.0037	1.0088	1.0140
set_diff	1.0132	1.0400	1.0676
p1_break_pt	0.9948	1.0936	1.2024
p2_break_pt	0.8100	0.8900	0.9780
roll_win_perc	1.0002	1.0031	1.0059
statusp1_break	0.9543	1.0134	1.0762
statusp2_break	0.9557	1.0146	1.0771
last_point_victor	0.9518	0.9978	1.0460
lastpt_p1_winner	0.9448	1.0017	1.0620
lastpt_p2_winner	0.9575	1.0142	1.0742
p1_serving_for_set	0.9187	1.0334	1.1624
p2_serving_for_set	0.8319	0.9327	1.0457
p1_serving_to_stay	0.8735	0.9562	1.0468
p2_serving_to_stay	0.9045	0.9912	1.0863

Interpret three of the above parameters. Which parameters have the largest confidence intervals? Which parameters' are you most uncertain about their effect on the point_victor?

Predictions

Using our full model, we can now predict the probability that a player will win a point in match given our set of predictors. We can also construct a confidence interval for this prediction by combining the variance of each of the parameters. Our estimate for the log-odds of Y:

- $\hat{g}(x) = \hat{\beta_0} + \hat{\beta_1} x_1 + ... + \hat{\beta_p} x_p$

And our confidence intervals of the log-odds:

- $\hat{g}(x) \pm z_{1-\frac{\alpha}{2}} \hat{SE}(\hat{g}(x))$

We can convert out log-odds into more probabilities for interpretation by:

- $\hat{p}(x) = \dfrac{e^{\hat{g}(x)}}{1 + e^{\hat{g}(x)}}$

Let's pick an interesting match to analyze as we assess our model's predictive abilities. We've compiled a list of the longest matches of Wimbledon 2022. The first ten are the longest men's matches and the second ten are the longest women's matches.

Table 10: Longest Wimbledon Matches

match_id	player1	player2	p1_pts_won	p2_pts_won	total_points
1403	David Goffin	Frances Tiafoe	205	198	403
1116	Jan-Lennard Struff	Carlos Alcaraz	188	192	380
1405	Cristian Garin	Alex de Minaur	180	185	365
1216	Jiri Vesely	Alejandro Davidovich Fokina	180	176	357
1145	Filip Krajinovic	Jiri Lehecka	171	177	348
1130	Adrian Mannarino	Max Purcell	177	168	345
1504	Taylor Fritz	Rafael Nadal	168	168	336
1132	Alejandro Davidovich Fokina	Hubert Hurkacz	166	168	334
1161	Lorenzo Sonego	Denis Kudla	171	157	328
1112	Enzo Couacaud	John Isner	155	170	325
2222	Panna Udvardy	Elise Mertens	132	133	265
2130	Serena Williams	Harmony Tan	124	119	243
2150	Daria Saville	Viktoriya Tomova	118	120	238
2147	Rebecca Marino	Katarzyna Kawa	109	108	217
2407	Tatjana Maria	Jelena Ostapenko	115	102	217
2504	Tatjana Maria	Jule Niemeier	109	97	206
2139	Caroline Garcia	Yuriko Miyazaki	107	98	205
2158	Tamara Korpatsch	Heather Watson	92	113	205
2231	Anhelina Kalinina	Lesia Tsurenko	100	105	205
2125	Coco Gauff	Elena-Gabriela Ruse	103	101	204

The longest match of Wimbledon 2022 featured Belgian David Goffin and American Frances Tiafoe. The data set designated Goffin as player 1 and Tiafoe as player 2. Goffin won a tightly contested match by three sets to two. The match lasted over four hours and 30 minutes. Let's pick five interesting points and calculate our model's predicted probability of victory for each of them. All probabilities will be from the perspective of Goffin, because he is player 1.

We've tabulated our five points with each of our parameters present.

Table 11: David Goffin vs. Frances Tiafoe: Predictor Values

point_no	rank_diff	server	pt_win_perc	set_diff
1	-30	0	50.000	0
106	-30	1	50.476	0
107	-30	1	50.943	1
270	-30	1	48.699	-1

Table 11: David Goffin vs. Frances Tiafoe: Predictor Values

point_no	rank_diff	server	pt_win_perc	set_diff
402	-30	0	50.623	0

We'll calculate the probability Goffin wins the first point "by hand". This table below will help us remember our model's estimates.

Table 12: Parameter Estimates

(Intercept)	rank_diff	server	pt_win_perc	set_diff
-1.1858	0.0007	1.0328	0.0134	0.0333

Log-odds of victory for Goffin in point 1:

- $\hat{g}(1)$ = -1.1858 + 0.0007(-30) + 1.0328(0) + 0.0134(50) + 0.0333(0) = -0.5368

Probability:

- $\hat{p}(x) = \frac{e^{-0.5368}}{1 + e^{-0.5368)}} = .3691$

Let's see what our predictions are for all five of our points.

Table 13: David Goffin vs. Frances Tiafoe: Predictions

point_no	log_odds	2.5%	prediction	97.5%	point_victor
1	-0.5361	0.3628	0.3691	0.3755	1
106	0.5030	0.6167	0.6232	0.6296	1
107	0.5426	0.6238	0.6324	0.6409	0
270	0.4460	0.6014	0.6097	0.6179	1
402	-0.5278	0.3647	0.3710	0.3774	1

Given our predictors, we estimate the probability of victory for Goffin in the first point is 36.91%. This probability of victory is as low as 36.28% and as high as 37.55% with 95% certainty.

In more statistical language, we are 95% confident that the true probability of winning the point given these parameters is between 36.28% and 37.55%.

is this right? ^

Pick a few points yourself to assess. We'll let you pick a match that interests you and a few point numbers. The first player listed in the title is player 1.

student selects model, match, and points they'd like to assess. I chose a model with every variable, if we are wanting a pdf version of this, we should select a model that does not go off the page lol

Below are the parameters of the 5 points you chose. Feel free to adjust your choices.

Table 14: Novak Djok

point_no	rank_diff	server	pt_win_perc	set_diff	p1_break_pt	p2_break_pt	roll_wi
51	37	0	46.000	0	0	0	4
100	37	1	49.495	-1	0	1	4
111	37	0	51.818	0	0	0	5
171	37	1	52.941	0	0	0	6
191	37	1	53.158	1	0	0	5

Given the values of the predictor variables at each of those points, below is the prediction table.

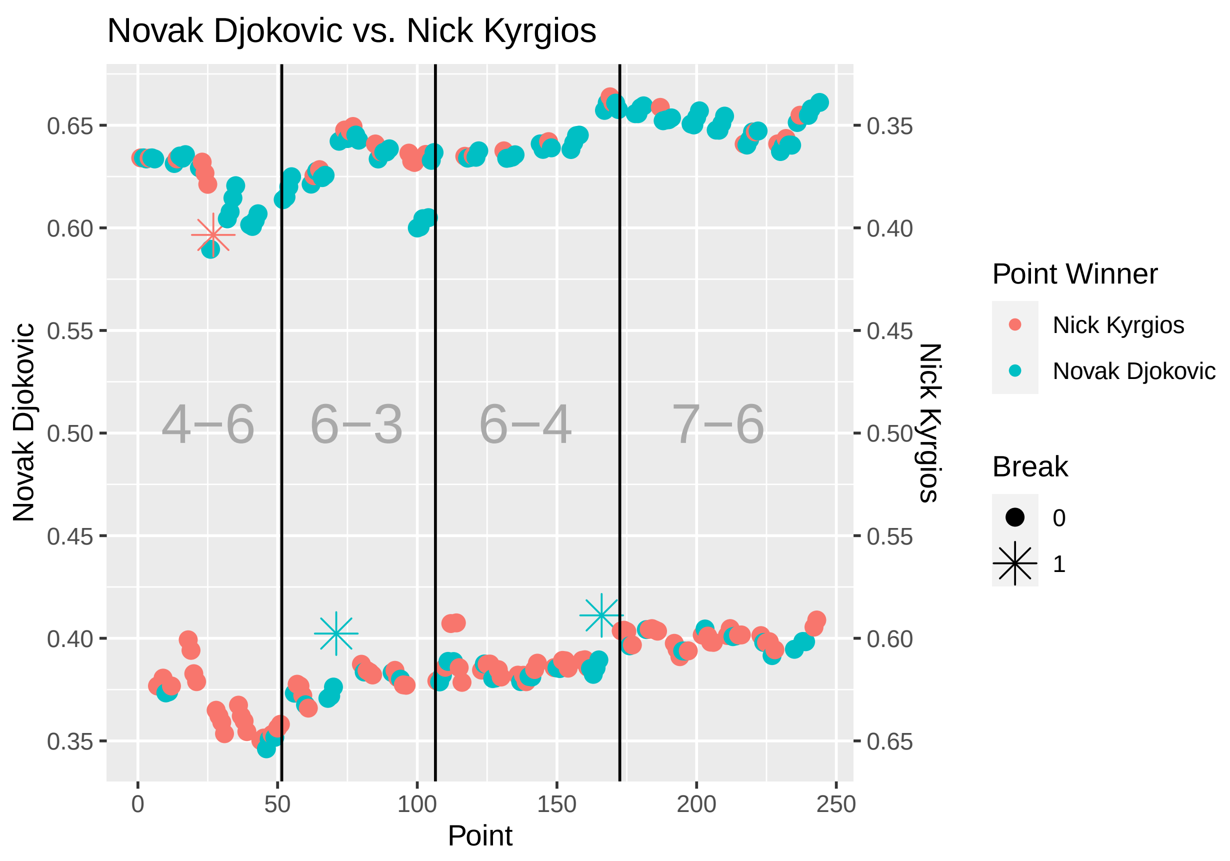
Table 15: Novak Djokovic vs. Nick Kyrgios: Predictions

point_no	log_odds	2.5%	prediction	97.5%	point_victor
51	-0.6897	0.3096	0.3341	0.3596	0
100	0.3529	0.5526	0.5873	0.6212	1
111	-0.6384	0.3328	0.3456	0.3586	1
171	0.4225	0.5756	0.6041	0.6319	1
191	0.4637	0.5994	0.6139	0.6282	1

Using the values from the table interpret at least three points. Which points did the model predict well? Which points did the model predict poorly?

In a table, we can only see a few of the points. Often, it can be helpful to visualize a lot of the model's predictions at once. We can plot the model's predictions of an entire match and compare the predictions with the actual results. The following plot visualizes an entire match at Wimbledon. Feel free to change the match to whichever match_id interests you. You can also adjust the model and see how the predictions change.

I really think this visualization below is the coolest thing ever. It is kinda tangentially applicable but so awesome



I think we might want to skip this part?

Attribution

Jeff Sackman's license: