Implementing Iterative Reweighted Least Squares

Ryker Dolese

Introduction

Iteratively reweighted least squares (IRLS) is a method used to solve certain optimization problems with constraints. It's commonly used in generalized linear models, including logistic and Poisson regression.

Here's a simplified explanation:

1. **Start with an initial estimate**: For logistic and Poisson regression, we start with an initial estimate of the parameters ( ). This could be a vector of zeros, which I have used in my implementation.
2. **Calculate the predicted values**: Using the current estimate of , we calculate the predicted values (π_i).
3. **Calculate the weights**: The weights are calculated based on the predicted values. In logistic regression, the weights are the predicted probabilities times (1 - predicted probabilities). In Poisson regression, the weights are just the predicted counts because the variance of a poisson variable is the mean itself. These are the respective variances for Bernoulli and Poisson random variables.
4. **Regress the adjusted response on the predictors using weights**: The adjusted response is the predictors times the current parameter estimates plus the residuals divided by the current predicted values, all divided by the square root of the weights. We then perform a weighted least squares regression of the adjusted response on the predictors using the weights.
5. **Update the estimates**: The coefficients from the weighted least squares regression are used to update the estimates of .
6. **Repeat steps 2-5**: These steps are repeated until the estimates converge (i.e., the estimates don't change much between iterations).

The IRLS method is used because it can handle the fact that the variance of the response depends on the predicted values, which is a characteristic of Poisson and logistic regression.

Implementing The Algorithm

```r
regression_irls <- function(x, y, n, max_iter = 1000, tol = 1e-6, type = "bernoulli") {

# Logistic Version of IRLS
irls_logistic <- function(x, y){
 # Initialize parameters
  beta <- rep(0, ncol(x))

  for (iter in 1:max_iter) {

    # Calculate predicted probabilities
    p <- 1 / (1 + exp(-x %*% beta))

    # Calculate working response and weights
    w <- p * (1 - p)

    #z <- X %*% beta + (y - p) / w
    eta <- x %*% beta
    z <- eta + (y - p)*(1/(p*(1-p)))
    W <- diag(as.vector(w))

    # Update parameters using weighted least squares
    beta_new <- solve(t(x) %*% W %*% x) %*% t(x) %*% t(W) %*% z

    # Check for convergence
    if (max(abs(beta_new - beta)) < tol) {
      break
    }

    # Update parameters for the next iteration
    beta <- beta_new
  }

  # redefine beta hat
beta_hat <- beta

# fitted values used to solve for variance and SE of estimators
fitted_values <- exp(x %*% beta_hat) / (1 + exp(x %*% beta_hat))
pi_hat <- fitted_values[,1]

# diagonal matrix of variances
V <- diag(pi_hat * (1 - pi_hat))

var_cov_beta_hat <- solve(t(x) %*% V %*% x)

std_error_beta_hat <- sqrt(diag(var_cov_beta_hat))

# calculate criteria for hypothesis testing of coefficient
z_values <- beta_hat/std_error_beta_hat
p_values <- 2*(1-pnorm(abs(z_values)))
# construct logistic regression table
coefficients <- paste0("Beta", 0:(length(beta_hat)-1))

# Construct logistic regression table
regression_results <- data.frame(
  Coefficients = coefficients,
  Estimate = beta_hat,
  Std_Error = std_error_beta_hat,
  z_value = z_values,
  p_value = p_values
)
#print(kable(regression_results, caption=""))

# Include a metric for Residual Deviance
```

```r
temp1 <- ifelse(y > 0, y*log(y/(1*pi_hat)), 0)
temp2 <- ifelse(y == 1, 0, (1-y)*log((1-y)/(1*(1-pi_hat))))
Deviance = 2*sum(temp1 + temp2)
cat("Residual Deviance: ", Deviance, " on ", length(y)-length(coefficients),
    " degrees of Freedom \n")
regression_results <<- regression_results

}



if (type == "bernoulli") {
 # algorithm is built for bernoulli data
irls_logistic(x = x, y = y)
  return(kable(regression_results))
}



# If type is binomial, we will use the above algorthm after transforming into bernoulli data
if (type == "binomial") {

### transform into bernoulli data first
convert_binomial_to_bernoulli <- function(x, y, n) {
  df_bernoulli <- data.frame()

  for(i in 1:length(y)) {
    # Create a temporary data frame with the replicated predictors
    df_temp <- data.frame(sapply(1:ncol(x), function(j) rep(x[i, j], n[i])))


    # Set the column names
    colnames(df_temp) <- paste0("x", 0:(ncol(x)-1))

    # Bind the Bernoulli response
    df_temp$y <- c(rep(1, y[i]), rep(0, n[i] - y[i]))

    # Append to the Bernoulli dataframe
    df_bernoulli <- rbind(df_bernoulli, df_temp)
  }

  return(df_bernoulli)
}

# Use the function on your matrix and vectors
df_bernoulli <- convert_binomial_to_bernoulli(x, y, n)

y <- df_bernoulli$y
x <- as.matrix(df_bernoulli[,-ncol(df_bernoulli)])

## now we can apply our same algoritm
irls_logistic(x = x, y = y)
```

```
}

# Poisson Process is much like the Bernoulli one, just changing our weights and variance estimates
irls_poisson <- function(x, y, max_iter = 100, tol = 1e-6) {
  # Initialize parameters
  beta <- rep(0, ncol(x))

  for (iter in 1:max_iter) {
    # Calculate predicted values
    p <- exp(x %*% beta)

    # Calculate working response and weights
    w <- p

    eta <- x %*% beta
    z <- eta + (y - p)*(1/p)
    W <- diag(as.vector(w))
    # Update parameters using weighted least squares
    beta_new <- solve(t(x) %*% W %*% x) %*% t(x) %*% t(W) %*% z

    # Check for convergence
    if (max(abs(beta_new - beta)) < tol) {
      break
    }

    # Update parameters for the next iteration
    beta <- beta_new

  }
  beta_hat <- beta
  #solve for fitted Values
fitted_values <- exp(x %*% beta)
var_cov_beta_hat <- solve(t(x) %*% W %*% x)

std_error_beta_hat <- sqrt(diag(var_cov_beta_hat))


z_values <- beta_hat/std_error_beta_hat
p_values <- 2*(1-pnorm(abs(z_values)))
# construct logistic regression table
coefficients <- paste0("Beta", 0:(length(beta_hat)-1))

# Construct logistic regression table
regression_results <- data.frame(
  Coefficients = coefficients,
  Estimate = beta_hat,
  Std_Error = std_error_beta_hat,
  z_value = z_values,
  p_value = p_values
)
#print(kable(regression_results, caption=""))

# Find residual deviance
```

```r
temp1 <- ifelse(y > 0, y*log(y/fitted_values), 0)
Deviance = 2*sum(temp1-(y - fitted_values))
cat("Residual Deviance: ", Deviance, " on ", length(y)-length(coefficients),
" degrees of Freedom \n")
regression_results <<- regression_results
#return(list(regression_results, Deviance))
}


##### If Poisson, use this function
if (type == "poisson") {
  irls_poisson(x=x,y=y)

}


}
```

Applications: Loan Acceptance

Dream Housing Finance is a company that specializes in all types of home loans and operates across urban, semi-urban, and rural areas. Initially, a customer applies for a home loan, after which the company verifies the customer's eligibility for the loan.

The company aims to streamline the loan eligibility process in real-time, using customer information provided during the online application process. This information includes Gender, Marital Status, Education, Number of Dependents, Income, Loan Amount, Credit History, among others. The company's goal is to automate this process by identifying segments of customers who are eligible for a loan, enabling them to focus their marketing efforts on these specific segments. They have provided a subset of their data for this purpose.

Our goal is to model this acceptance.

1. **`Loan_ID`**: A character vector representing the unique ID for each loan.
2. **`Gender`**: A character vector indicating the gender of the applicant. It can be 'Male' or 'Female'.
3. **`Married`**: A character vector indicating the marital status of the applicant. It can be 'Yes' or 'No'.
4. **`Dependents`**: A character vector indicating the number of dependents of the applicant.
5. **`Education`**: A character vector indicating the education level of the applicant. It can be 'Graduate' or 'Not Graduate'.
6. **`Self_Employed`**: A character vector indicating whether the applicant is self-employed or not. It can be 'Yes' or 'No'.
7. **`ApplicantIncome`**: An integer vector representing the income of the applicant.
8. **`CoapplicantIncome`**: An integer vector representing the income of the co-applicant.
9. **`LoanAmount`**: An integer vector representing the loan amount in thousands.
10. **`Loan_Amount_Term`**: An integer vector representing the term of loan in months.
11. **`Credit_History`**: An integer vector indicating whether the applicant has a credit history or not. It can be 1 (Yes) or 0 (No).
12. **`Property_Area`**: A character vector indicating the area of the property. It can be 'Urban', 'Semiurban', or 'Rural'.

13. `Loan_Status`: whether they were approved for the loan (Y/N)

Exploratory Data Analysis

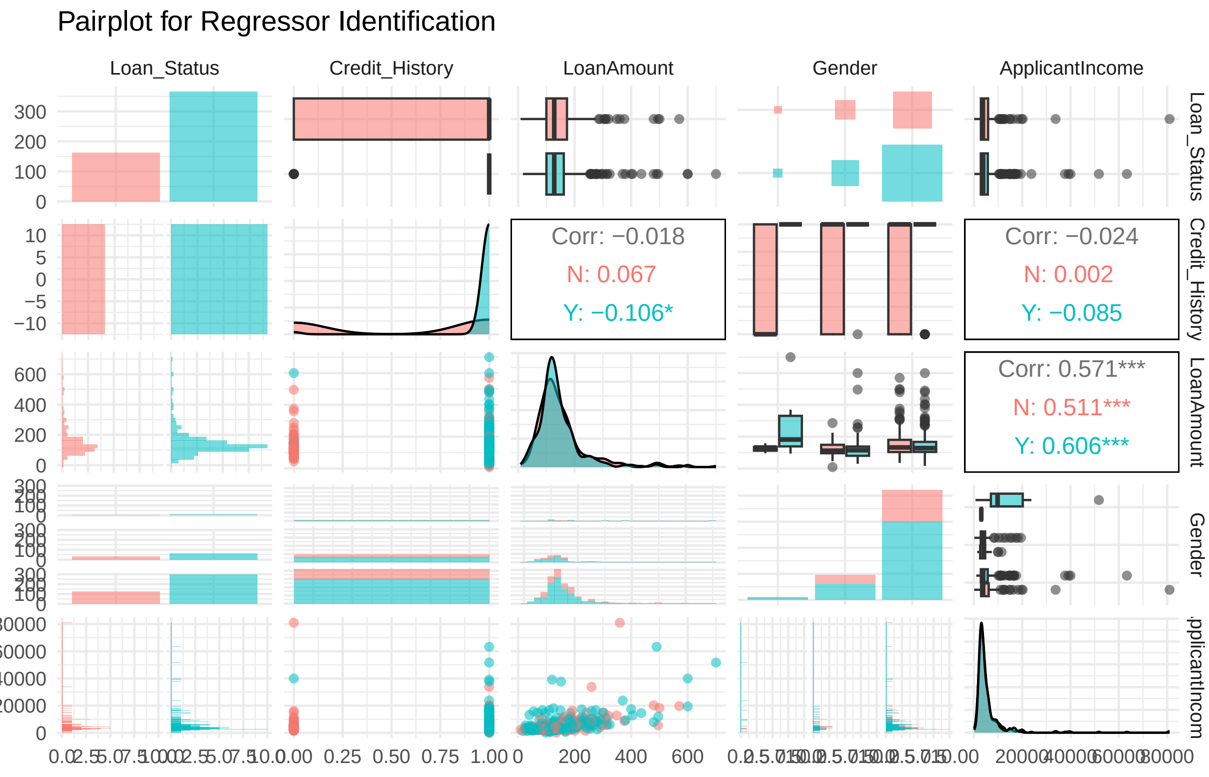

Figure 1: Regressor Pairplot

Based on our pairplot (Figure 1), it appears that Loan Status could be highly related to Credit History as well as gender. While their aren't many observations with gender "other," these individuals tend to be denied loans more frequently. This is something to consider when modeling. Applicant Income and Loan Amount seem to be very correlated, but don't have much to say about Loan Status. For this reason, we might want to try Credit History as our first predictor.

Simple Logistic Regression Modeling: One Predictor

```
Residual Deviance:  499.3183  on  524  degrees of Freedom
```

Table 1: Logistic Regression Results

	Coefficients	Estimate	Std_Error	z_value	p_value
	Beta0	-2.316770	0.3961586	-5.848086	0
Credit_History	Beta1	3.683646	0.4131970	8.914987	0

While this model is extremely simple (we have used a factor – credit history – as the only predictor), it actually tends to model the data pretty well, as seen in Figure 2 and Table 1. We have opted to try a new model with some additional predictors to see if the prediction power increases. The results can be seen in Table 2.

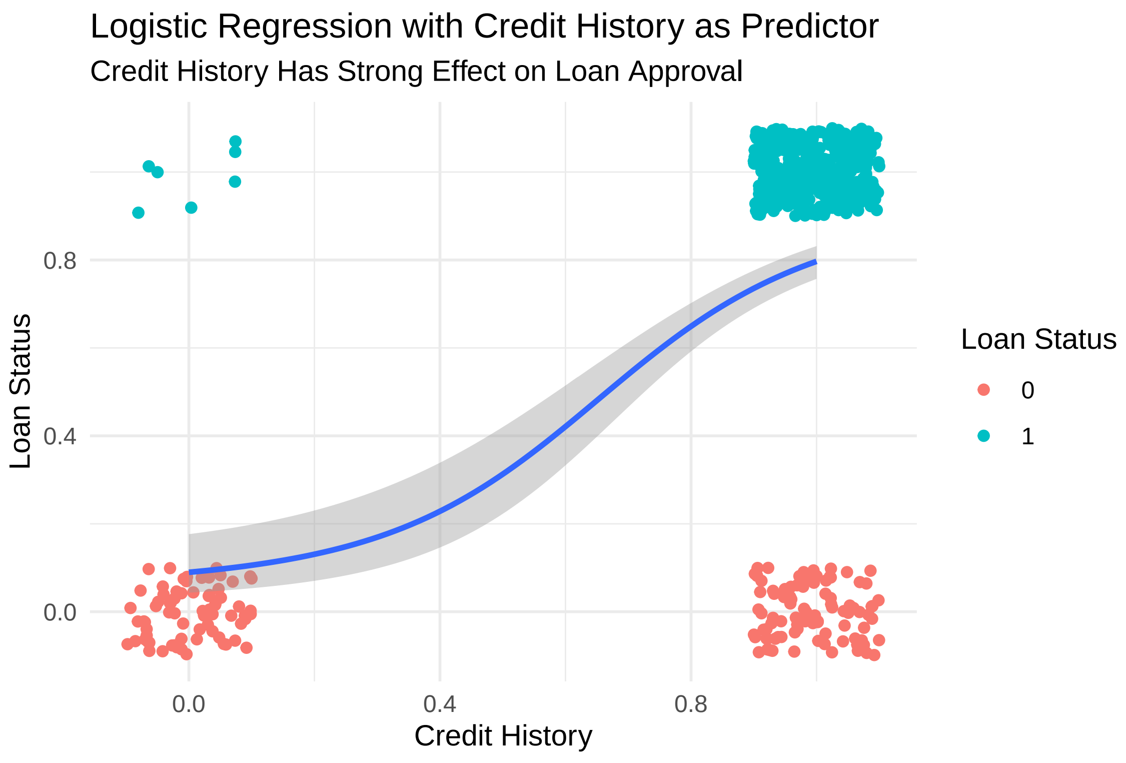

Figure 2: Simple Logistic Regression

```
Residual Deviance:  496.3757  on  522  degrees of Freedom
```

Table 2: Multiple Logistic Regression Results

	Coefficients	Estimate	Std_Error	z_value	p_value
	Beta0	-2.4002536	0.4867531	-4.931152	0.0000008
Credit_History	Beta1	3.6897632	0.4141489	8.909267	0.0000000
LoanAmount	Beta2	-0.0015284	0.0013685	-1.116805	0.2640778
Gender	Beta3	0.3812928	0.2708117	1.407963	0.1591420

A new model with additional predictors LoanAmount, Gender, Loan_Amount_Term yield a slightly stronger model with a residual deviance of 496. However, this may have come at the cost of overfitting the data.

The Binomial Case: Working With Proportions

At the group level, we look at the proportion of successes and failures (or male/famale, tall/short, dem/rep, ...) in groups/populations/sites. In all these cases, we know both numbers ... those of `successes` and those of `failures`, in contrast to Poisson regression where we just have counts of `successes`.

In the past, the way this data was modelled was to use a single percentage as the response variable. However, this causes problems because:

- The errors are not normally distributed.
- The variance is not constant.
- The response is bounded (by 1 above and by 0 below).
- We lose information on the size of the sample, n, from which the proportion was estimated.

In the GLM framework, we can model proportion data directly. R carries out weighted regression, using the individual sample sizes as weights, and the logit link function to ensure linearity.

In this example, we have a set of data on the performance of cars in terms of fuel efficiency. In summary form this provides a set of total counts and poor performance counts for car models that fall into 12 weight bands.

```
Residual Deviance:  187.5987  on  324  degrees of Freedom
```

Table 3: Binomial Logistic Regression Results

	Coefficients	Estimate	Std_Error	z_value	p_value
x0	Beta0	-13.7109822	1.4423435	-9.506045	0
x1	Beta1	0.0043048	0.0004604	9.349644	0

Here, our model appears to be especially robust. Our residual deviance is far greater than our degrees of freedom, suggesting weight is a strong predictor of poor fuel efficiency in cars.

Something to note, however, is the manner in which logistic regression is performed on these proportions. Rather than giving each proportion the same weight, the algorithm weighs each observation based on the total number of observations for the given x value. For instance, there were 48 observations with a weight of 2100 lbs, so this observations bears more weight in the model outcome. As seen in figure 3, I have emphasized this notion with the size of each point; the larger sizes imply larger weights.

Working with Count Data: Poisson Regression

Applications: PhD Publications

Dataset Description: Doctoral Publications in Biochemistry

The Doctoral Publications dataset offers a glimpse into the scientific productivity of PhD students specializing in biochemistry. Here are the key details:

1. Objective: Understand the research output of biochemistry PhD students during the final three years of their doctoral programs.
2. Variables:

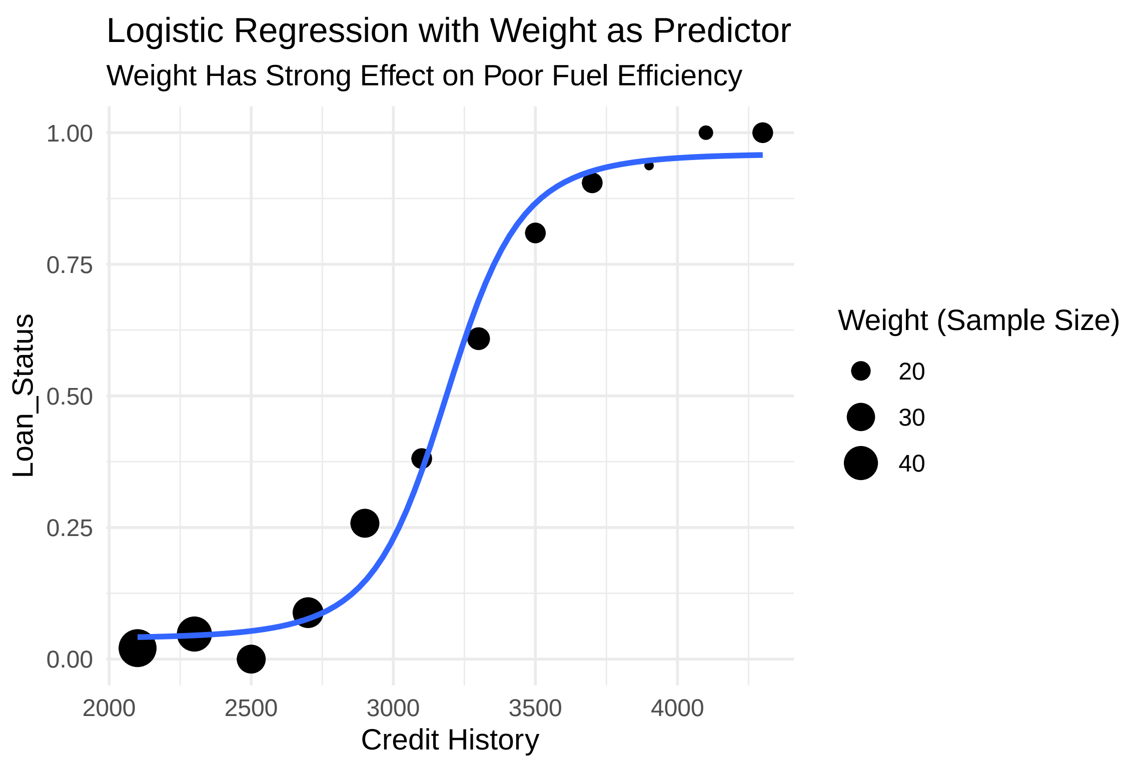

Figure 3: Logistic Regression on Proportions

- Articles: The number of scholarly articles published by each student within the specified time-frame.
- Gender: Categorized as male or female.
- Marital Status: Indicates whether the student is married.
- Children: The count of children aged less than six years.
- Program Prestige: Reflects the prestige level of the graduate program (e.g., program ranking).
- Mentor's Influence: The number of articles published by the student's mentor (supervisor).

For this implementation of the IRLS algorithm, we will be attempting to model articles (the number of publications for PHD students). This is a count variable, so poisson regression could be helpful for modeling. The first intuition could be to see whether there exists a relationship between the number of mentor publications and the number of articles published by their disciples.

The outcome of this regression can be seen in Table 4.

```
Residual Deviance:  662.8051  on  638  degrees of Freedom
```

Table 4: Poisson Regression Results

	Coefficients	Estimate	Std_Error	z_value	p_value
	Beta0	0.7187911	0.0354263	20.289741	0
mentor	Beta1	0.0149489	0.0020463	7.305139	0

Honestly, the model appears to perform well. The p-value is nearly zero for our "mentor" coefficient and the degrees of freedom close to the calculated residual deviance.

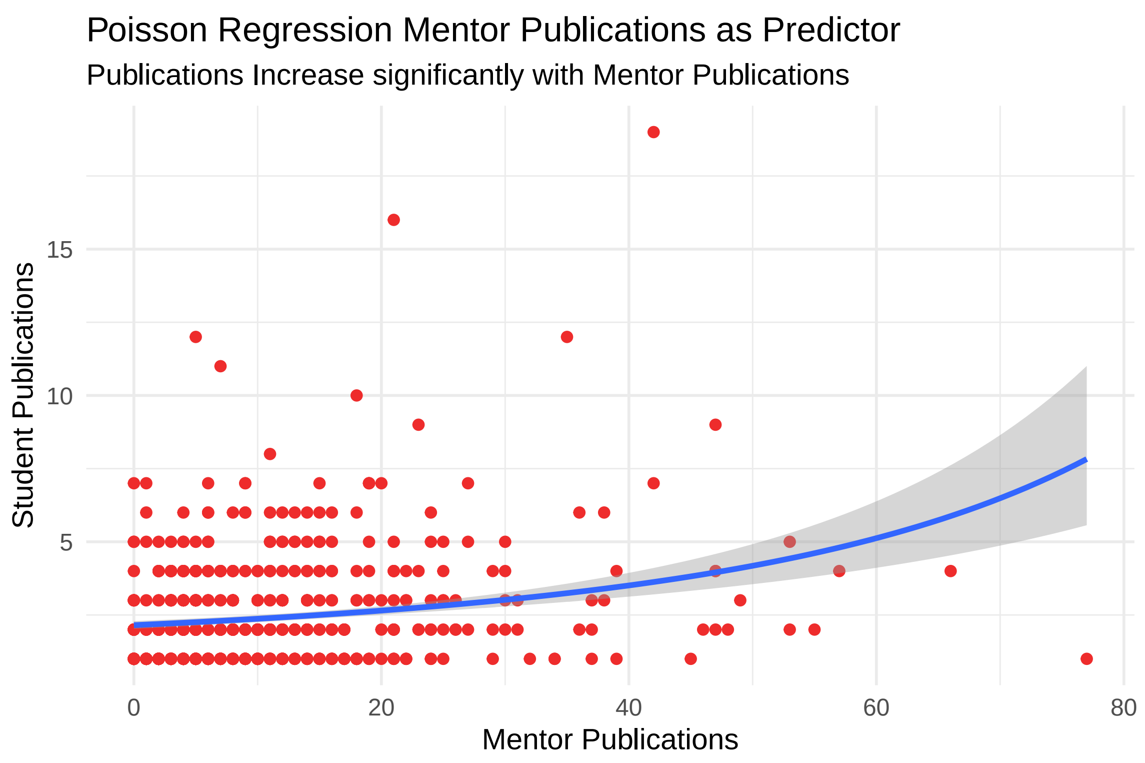

Figure 4: Simple Poisson Regression

In the pair plot (Figure 5), there aren't many other variables that stand out, especially with the numerical variables. We might want to consider adding some factor variables such as gender. These seem like they could have an effect on our model.

```
Residual Deviance:  655.6556  on  637  degrees of Freedom
```

Table 5: Multiple Poisson Regression Results

	Coefficients	Estimate	Std_Error	z_value	p_value
	Beta0	0.9225008	0.0834355	11.056454	0.0000000
mentor	Beta1	0.0143670	0.0020515	7.003351	0.0000000
gender	Beta2	-0.1388834	0.0521544	-2.662928	0.0077464

We have refit the model, and the results are shown in Table 5. Our model has improved slightly. The deviance fell by a greater margin than our degrees of freedom and the gender coefficient has a p-value far less than a 0.05 threshold.

Because we have only implemented a new factor variable, we can still visualize our model in 2 dimensions. This can be seen in Figure 6. The new model shows that males tend to have an exponential increase in publications as the number of mentors increase while females tend to have a relatively constant rate of student publications.

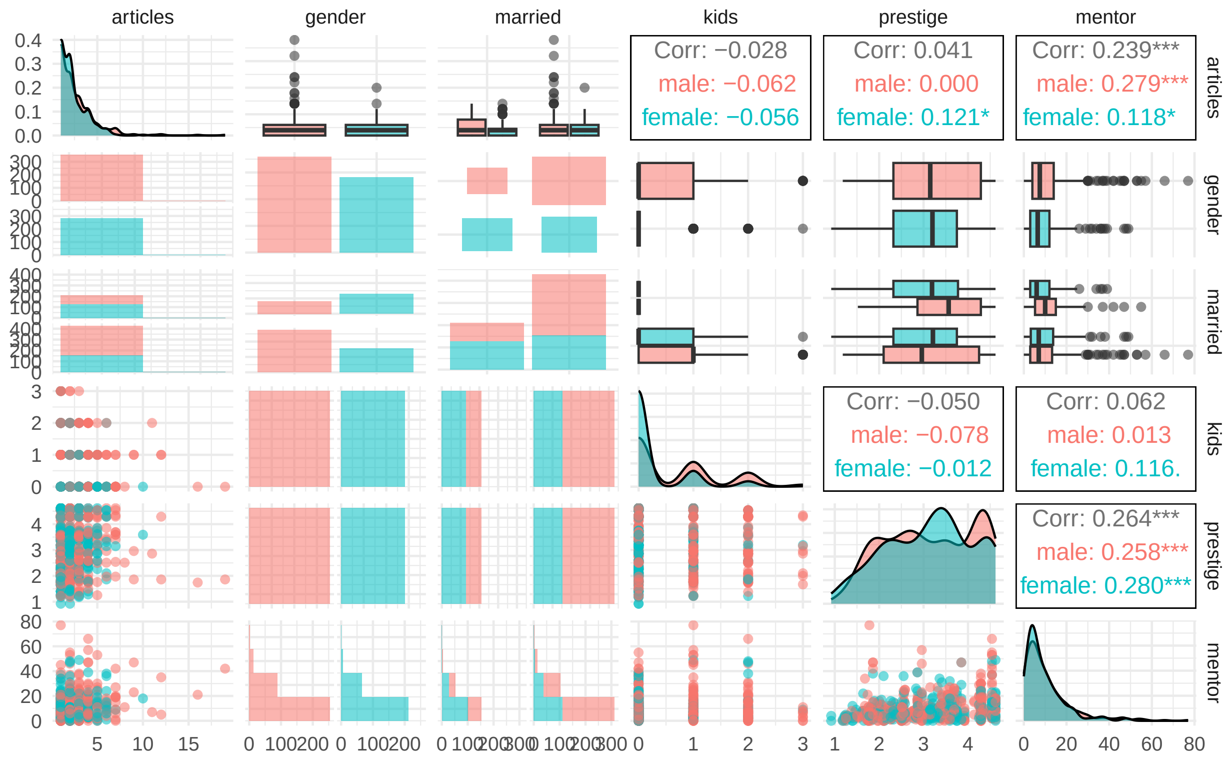

Figure 5: Pairplot for Poisson Regressors

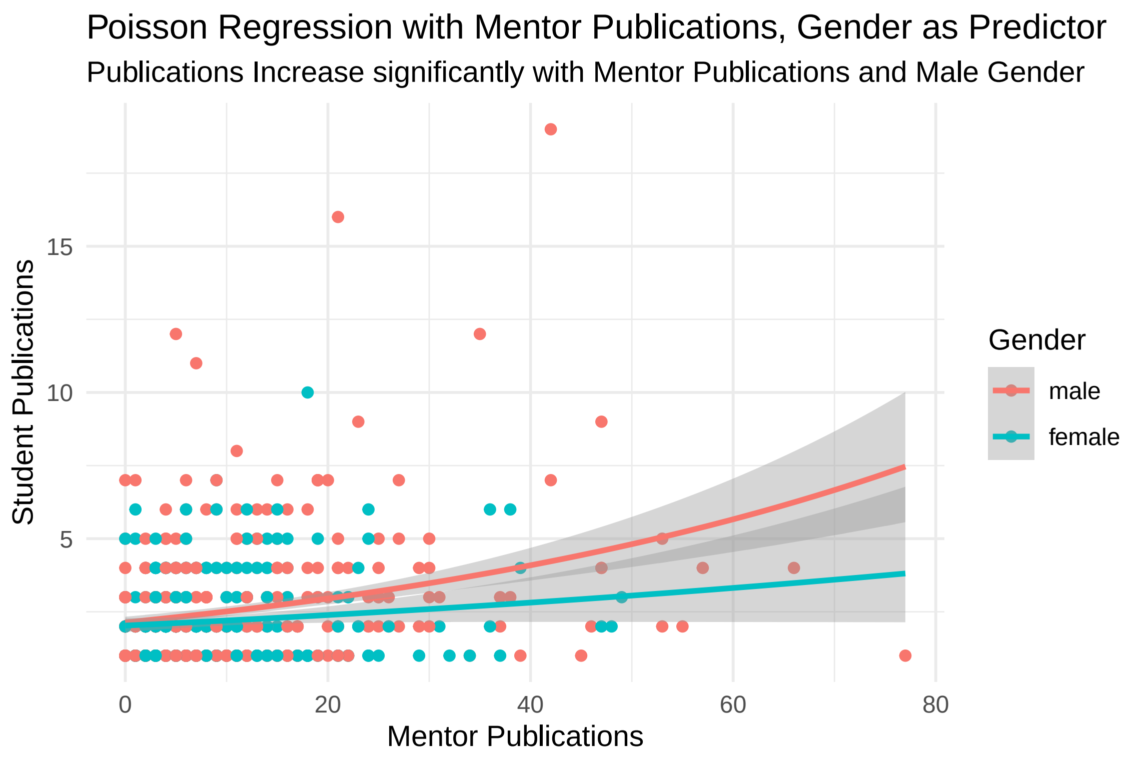

Figure 6: Multiple Poisson Regression

Modeling Multinomial Data: An Extension of Logistic Regression and IRLS

Multinomial Logistic Regression is a powerful classification technique that extends the standard logistic regression algorithm to handle scenarios with multiple possible outcome classes. Let's delve into its key aspects:

1. Problem Setting:
   - In many real-world scenarios, we encounter categorical dependent variables with two or more unordered levels (such as different flower species, customer preferences, or job roles).
   - Unlike binary logistic regression, where we predict a single outcome (e.g., yes/no), multinomial logistic regression deals with multiple possible outcomes.
   - For instance, consider predicting a child's food choice based on their parents' preferences and hobbies. The food choices (fast food, healthy options, vegan, etc.) represent the different levels of the dependent variable.
2. Model Mechanics:
   - Multinomial logistic regression estimates the probabilities of each category (outcome class) using a linear combination of the observed features (independent variables) and specific model parameters.
   - The model computes the log-odds (logarithm of the odds) for each category relative to a chosen reference category.
   - The probabilities are then derived from these log-odds using the softmax function.
   - Essentially, it's like running individual binomial logistic regressions for each category (except the reference category) simultaneously.
3. Interpretation and Output:
   - The output provides coefficients for each independent variable, just like in other regression models.
   - The relative probabilities of being in any non-reference category are compared to the reference category.
   - Researchers often choose the most common or frequent category as the reference.
   - Assumptions include independence of cases, correct model specification, and no multicollinearity among independent variables

The Multinomial IRLS Algorithm

```r
irls_multinomial <- function(x, y, max_iter = 1000, tol = 1e-6) {
  n_rows_x <- nrow(x)
  n_cols_x <- ncol(x)
  unique_y <- unique(y)
  n_unique_y <- length(unique_y)

  # Initialize beta vectors for each additional value of y (excluding the base value)
  beta_matrix <- matrix(0, nrow = n_cols_x, ncol = n_unique_y)

  for (iter in 1:max_iter) {
    p <- matrix(0, nrow = n_rows_x, ncol = n_unique_y)
    w <- matrix(0, nrow = n_rows_x, ncol = n_unique_y)
    eta <- matrix(0, nrow = n_rows_x, ncol = n_unique_y)
```

```r
      z <- matrix(0, nrow = n_rows_x, ncol = n_unique_y)

      exp_beta_x <- exp(x %*% beta_matrix)
      sum_exp_betax <- rowSums(exp_beta_x)

      # Calculate predicted probabilities for each category
      for (i in 1:(n_unique_y - 1)) {
        p[, i] <- exp_beta_x[, i] / (1 + sum_exp_betax)

        # Calculate working response and weights
        w[, i] <- p[, i] * (1 - p[, i])
        eta[, i] <- x %*% beta_matrix[, i]
        z[, i] <- eta[, i] + (y == unique_y[i]) - p[, i]

        # Update parameters using weighted least squares
        W <- diag(w[, i])
        beta_new <- solve(t(x) %*% W %*% x) %*% t(x) %*% W %*% z[, i]

        # Check for convergence
        if (max(abs(beta_new - beta_matrix[, i])) < tol) {
          break
        }

        # Update parameters for the next iteration
        beta_matrix[, i] <- beta_new
      }
    }

    # Calculate probabilities for each category (fitted values)
    p <- exp(x %*% beta_matrix)
    p <<- p / rowSums(p)

    beta_df <- data.frame(beta_matrix)
    colnames(beta_df) <- c("Intercept", "X1", "")
    return(kable(beta_df[,-ncol(beta_matrix)], caption = "Multinomial Regression Output"))
  }
```

Applications: Iris Data

The Iris dataset is a well-known benchmark in machine learning and statistics. It contains measurements of 150 iris flowers from three different species: Setosa, Versicolor, and Virginica. Each flower is characterized by four features:

1. Sepal Length
2. Sepal Width
3. Petal Length
4. Petal Width

The goal is to predict the species of the iris flower based on these features. The dataset serves as an excellent example for demonstrating multinomial logistic regression due to its multiple outcome classes.

To see how this could work, we have made a simple plot of sepal and petal lengths and can observe how the flower species form groups (Figure 7) . We can use this information in a classification model.

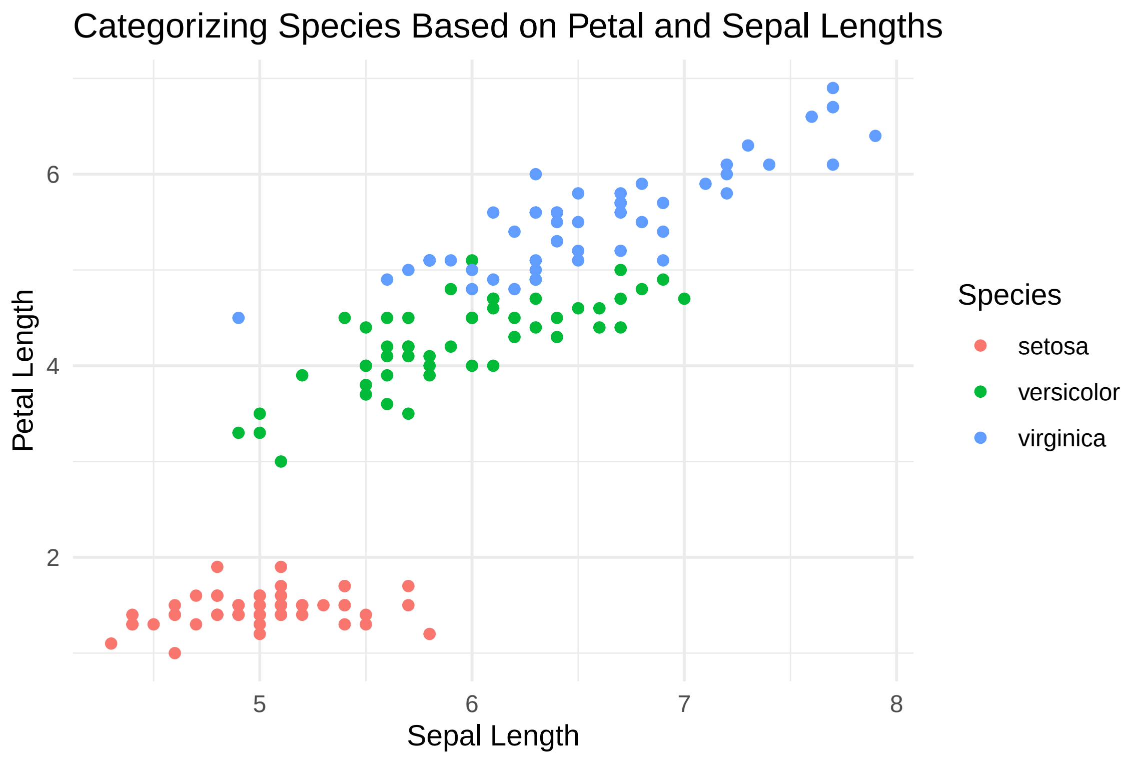

Figure 7: Visualizing the Iris Dataset

For our model, we have chosen sepal length as a predictor. Although this might not be the best regressor variable, it is useful in showing how categorical probabilities change across different lengths.

Table 6: Multinomial Regression Output

Intercept	X1
42.490971	14.340031
-7.346318	-2.178466

In table 7, we can see our multinomial coefficients. There are 4 values: 2 for the different intercepts and 2 for the sepal length predictor. These are then compared with the base outcome to determine probability using the softmax function. A plot of the predicted values can be seen in Figure 8. As one can see, our model predicts a lower probability for species setosa as sepal length increases. Virginica appears to have the highest probability when sepals are especially long while versicolor lies somewhere in the middle. The faceting makes this extremely apparent.

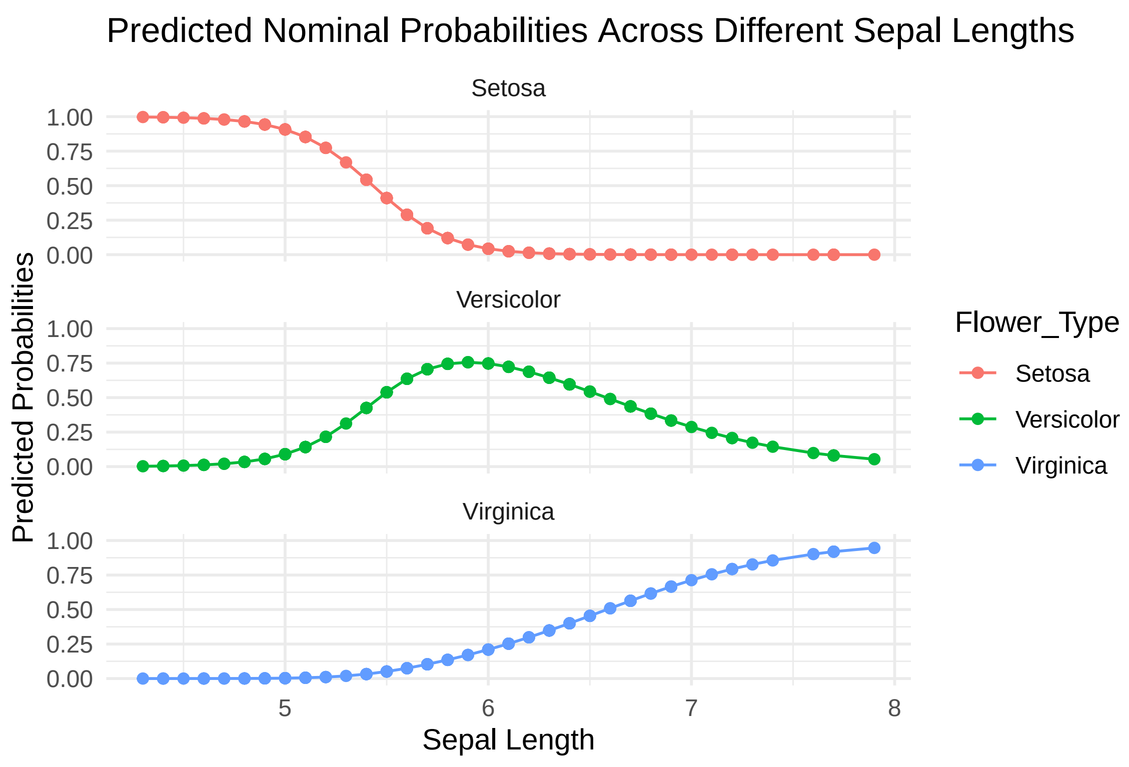

Figure 8: Multinomial Logistic Regression

Conclusion

In our project, we explored how the IRLS algorithm adapts to different scenarios, showcasing its robustness and flexibility. Here's a summary of our findings:

When dealing with binary outcomes (yes/no, success/failure), IRLS shines. It efficiently estimates the logistic regression coefficients, allowing us to predict probabilities and make informed decisions. Whether it's predicting customer churn, click-through rates, or disease diagnosis, IRLS handles Bernoulli data effectively. For proportion data (e.g., number of successes out of a fixed number of trials), IRLS extends naturally. Think of scenarios like website conversions, where we track the number of clicks out of total visits. By modeling the log-odds of success, IRLS accommodates overdispersion and provides accurate parameter estimates.

In cases involving count data with a known exposure (e.g., events per unit time), Poisson regression using IRLS is our go-to. Examples include accident rates, call center queries, or rare disease occurrences. IRLS handles the Poisson distribution gracefully, capturing the relationship between predictors and event rates. The true strength of IRLS emerges when dealing with multiple unordered outcome classes. Whether it's iris flower species, customer preferences, or job roles, multinomial logistic regression using IRLS provides accurate probabilities.

By iteratively updating parameters, IRLS ensures convergence and efficient computation.

Appendix:

https://www.kaggle.com/datasets/vikasukani/loan-eligible-dataset?resource=download

https://rdrr.io/cran/AER/man/PhDPublications.html

Link to Code