Project Report: Analysis of Bird Population Dynamics

Introduction

The study aims to analyze the population dynamics of birds at Jasper Ridge, focusing on understanding the temporal and spatial patterns in bird sightings. The analysis encompasses exploratory data analysis (EDA) to understand the distribution of bird sightings, land cover changes over time, and both temporal and spatial autocorrelation checks. Furthermore, predictive models including Generalized Linear Models (GLMs), Ridge regression, and Lasso regression are developed to predict bird counts for each season.

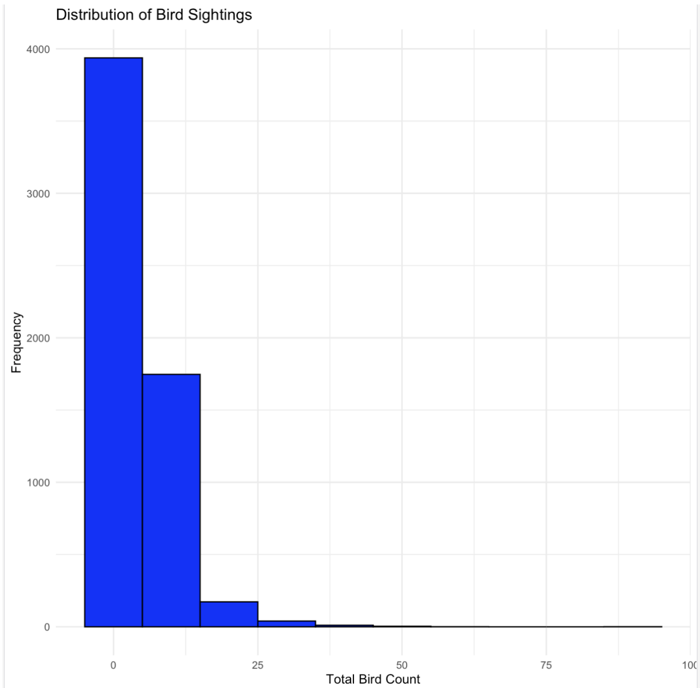
Data Preparation

The dataset used in this study includes bird sighting data collected at Jasper Ridge. The dataset is preprocessed by aggregating the counts of bird sightings to obtain the total bird count. The Date variable is converted to the proper date format, and Month and Year variables are extracted. The data are then organized by season, dividing the observations into Spring, Summer, Fall, and Winter datasets.

Exploratory Data Analysis (EDA)

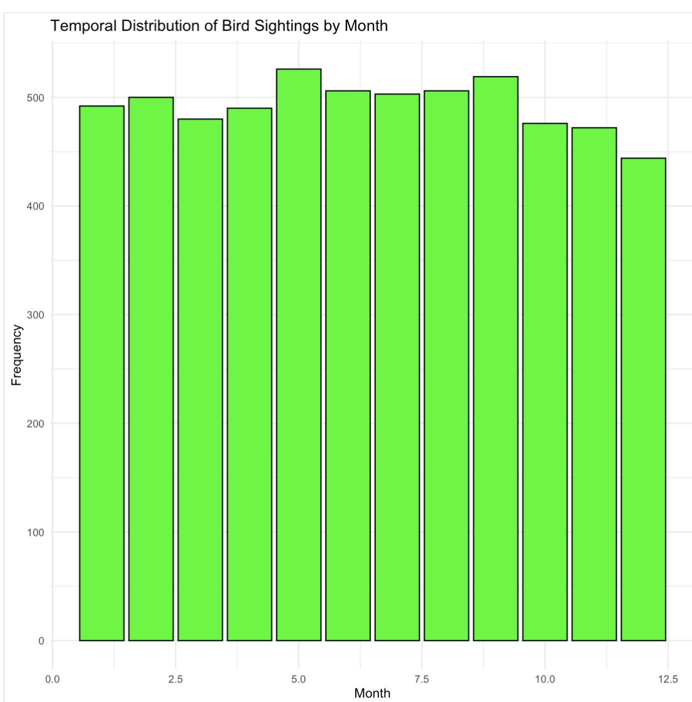
The exploratory data analysis reveals insights into the distribution of bird sightings and the temporal distribution of sightings by month. The histograms of bird sightings count show left-skewed distributions (Figure 1). Given the count nature of the data, the assumption of normality is not necessary due to the Poisson distribution nature of the count data.

A



Distribution of Bird Sightings
4000
3000
2000
1000
0
Frequency
0
25
50
75
100
Total Bird Count

B

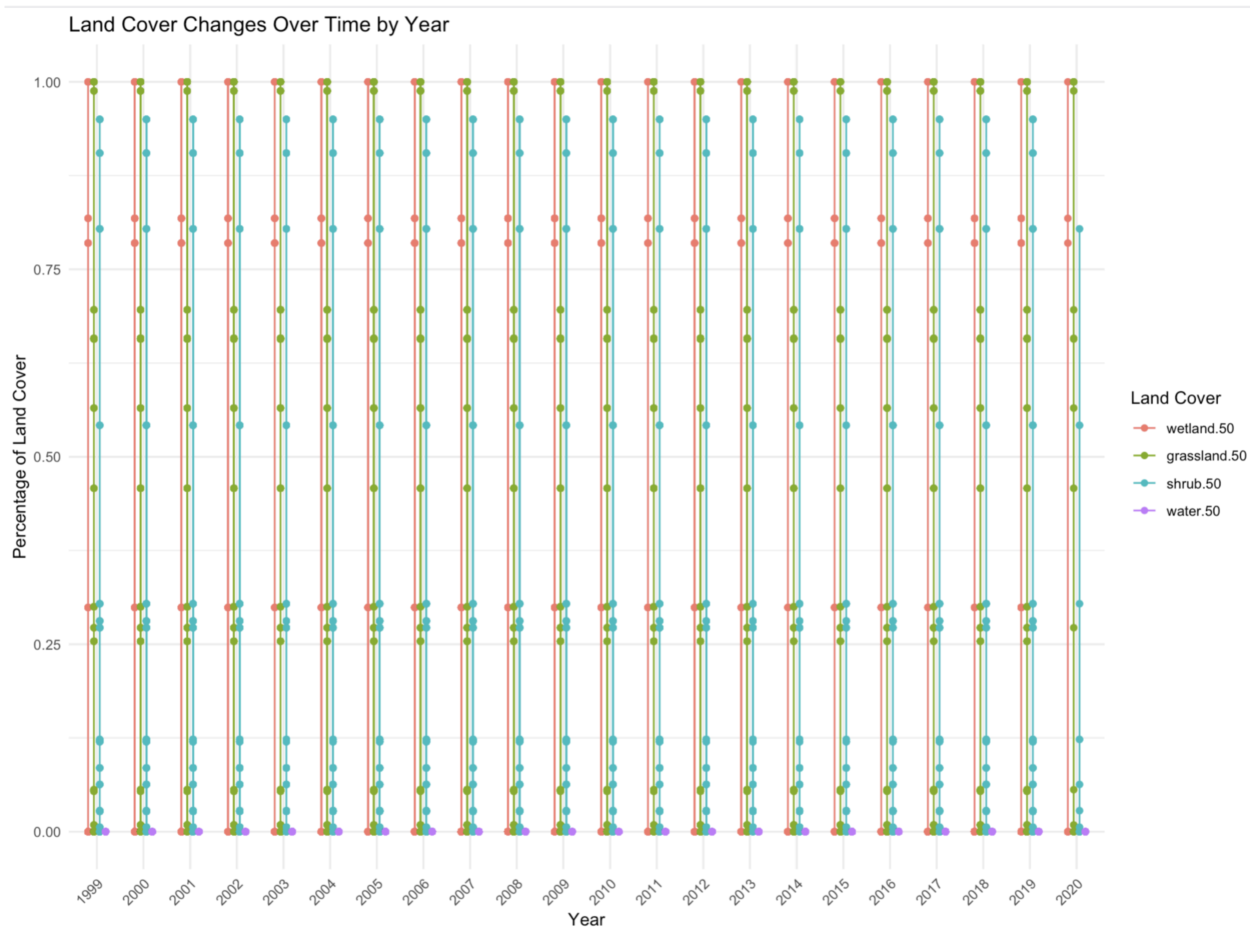


Temporal Distribution of Bird Sightings by Month
500
400
300
200
100
0
Frequency
0.0
2.5
5.0
7.5
10.0
12.5
Month

Figure 1: Frequency distribution of bird sightings. (A) shows the frequency of bird count when bird sighting and (B) shows the temporal distribution of bird sightings by month. This data is from 1999-2020 from Jasper Ridge.

Additionally, the land cover changes over time are visualized to understand the alterations in different land cover types over the years (Figure 2). Plot B in Figure 2 is representative of the cyclical changes that occur, causing changes in land cover percentage. Water cover remains stable across the entire year of 2020, but others fluctuate. These plots are included to provide insights into the environmental changes that might influence bird population dynamics.

A



B

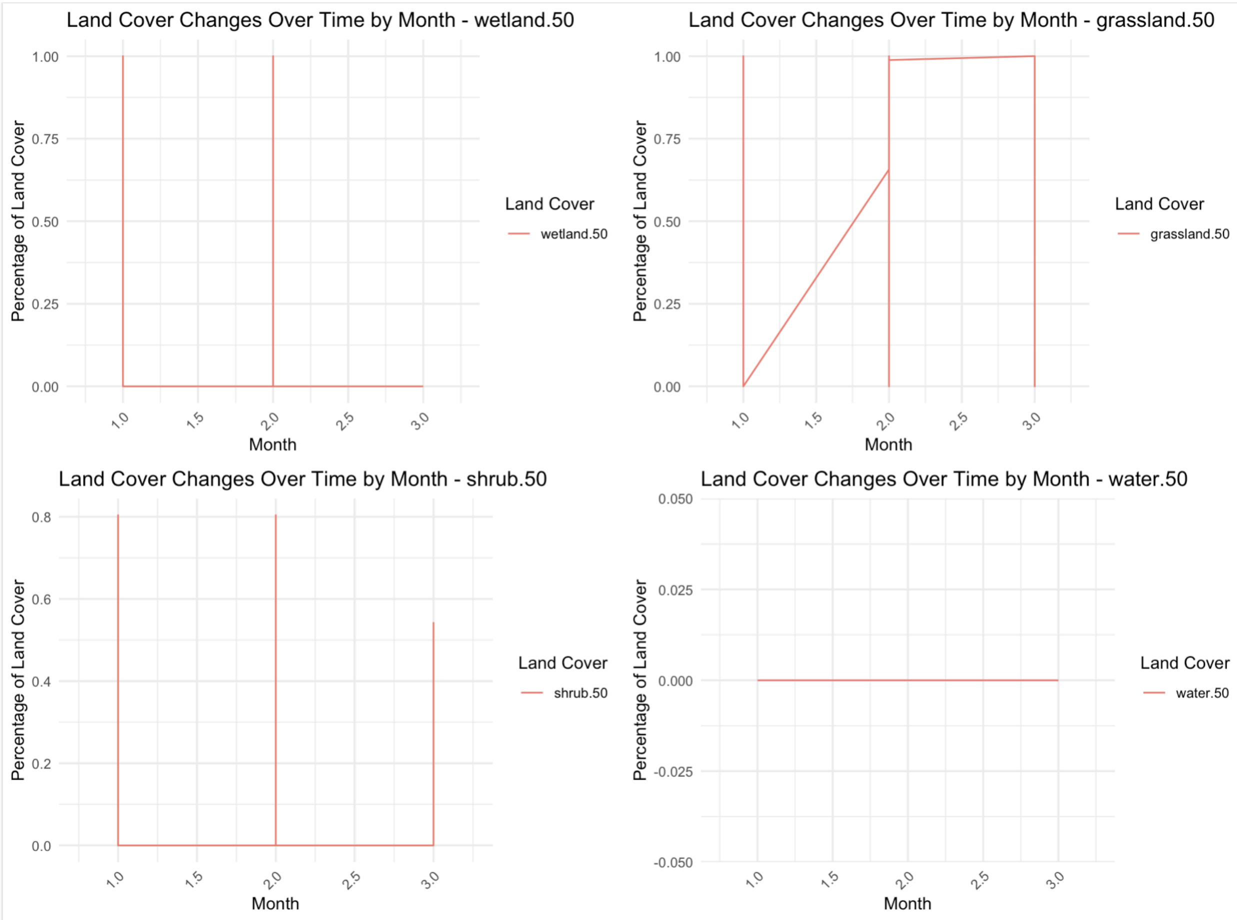


Figure 2: Land cover changes over time. (A) shows the changes in land cover from the years 1999-2020. The scatter plots connect the recorded percent land cover that each land cover type occupies in a 50-meter radius around the bird sighting. The legend on the right shows the different types of land cover. (B) is the land cover change for the year 2020. Each of the land cover type is split, such that 4 subgraphs are shown.

Spatial Autocorrelation Check

Moran's I test is performed to assess spatial autocorrelation (Figure 3). The test results indicate a strong positive spatial autocorrelation in the total bird count data ($p<0.05$, so null hypothesis is rejected), implying that areas with similar total bird counts tend to be clustered together spatially. To address spatial autocorrelation, spatial filters are added to the GLM models.

```
        Moran I test under randomisation

data:  bird_data$n.total
weights: bird_data.listw

Moran I statistic standard deviate = 22.853, p-value < 2.2e-16
alternative hypothesis: greater
sample estimates:
Moran I statistic       Expectation          Variance
     1.319538e-01     -1.691189e-04      3.342472e-05
```

Figure 3: Results from Morans test for spatial autocorrelation.

Spatial filters are used to account for the spatial autocorrelation present in the data. Spatial autocorrelation occurs when nearby locations tend to have similar values compared to locations that are farther apart. In the case of bird population data, this would imply that areas close to each other are likely to have similar bird counts due to factors such as habitat suitability, food availability, and ecological conditions. In the analysis, spatial filters were applied to address spatial autocorrelation in the total bird count data. This was done by adding lagged values of land cover types as spatial filters in the Generalized Linear Models (GLMs). These spatial filters capture the spatial dependence between neighboring locations and help improve the accuracy of the models by accounting for the spatial autocorrelation present in the data.

For each land cover type (e.g., wetland, grassland, shrub, water), lagged values were calculated using a spatial weights matrix. The spatial weights matrix was constructed based on the spatial proximity of observations using the k-nearest neighbor (kNN) approach. The k value was set to 5 because there are 5 major land cover types including 'Mixed'. This matrix represents the spatial relationships between different locations in the study area.

Temporal Autocorrelation Check

Generalized Least Squares (GLS) models are employed to assess temporal autocorrelation. The analysis reveals a significant relationship between the total bird count and time (month), considering the autocorrelation within each month (Figure 4). The estimated value of the AR(1) parameter, which indicates the correlation between successive observations within the same month. A Phi of 0.1654994 suggests a moderate positive correlation. Temporal autocorrelation is further addressed by considering the data in seasonal form, and dividing the observations into Spring, Summer, Fall, and Winter datasets.

```
Generalized least squares fit by REML
  Model: n.total ~ 1
  Data: bird_data
  Log-restricted-likelihood: -18123.83

Coefficients:
(Intercept)
   4.992926

Correlation Structure: AR(1)
 Formula: ~1 | Month
 Parameter estimate(s):
      Phi
0.1654994
Degrees of freedom: 5914 total; 5913 residual
Residual standard error: 5.255619
```

Figure 4: Results of the GLS test

Predictive Modeling

Predictive models are developed using Generalized Linear Models (GLMs), Ridge regression, and Lasso regression for each season. The models utilize lag variables of land cover types as predictors to forecast bird counts. The Mean Absolute Error (MAE) is calculated for each model to evaluate predictive performance.

GLMs: Generalized Linear Models are fitted for each season. The models suggest a significant relationship between bird counts and lag variables of land cover types, capturing both spatial and temporal effects.

Ridge Regression: Ridge regression models are fitted for each season. The models utilize a penalty term to reduce overfitting and improve predictive performance.

Lasso Regression: Lasso regression models are fitted for each season. The models perform variable selection and regularization to enhance predictive accuracy.

It is important to mention that linear regression was not used in this analysis because the data represents count data, which is more appropriately modeled using a Poisson distribution. Count data, such as the number of birds, typically exhibit a Poisson distribution due to the nature of the data being non-negative integers. Therefore, models like GLM, Ridge Regression, and Lasso Regression, which can handle Poisson-distributed data, were chosen to ensure the accuracy and appropriateness of the predictions.

Results

he provided results show the mean absolute error (MAE) for three different regression models—Generalized Linear Model (GLM), Ridge Regression, and Lasso Regression—across four seasons: Spring, Summer, Fall, and Winter.

For the GLM:

- The MAE for Spring is 3.550265.
- The MAE for Summer is 3.373538.
- The MAE for Fall is 3.556377.
- The MAE for Winter is 3.180541.

For Ridge Regression:

- The MAE for Spring is 3.552154.
- The MAE for Summer is 3.379585.
- The MAE for Fall is 3.563616.
- The MAE for Winter is 3.204355.

For Lasso Regression:

- The MAE for Spring is 3.549097.
- The MAE for Summer is 3.372767.
- The MAE for Fall is 3.564943.
- The MAE for Winter is 3.191702.

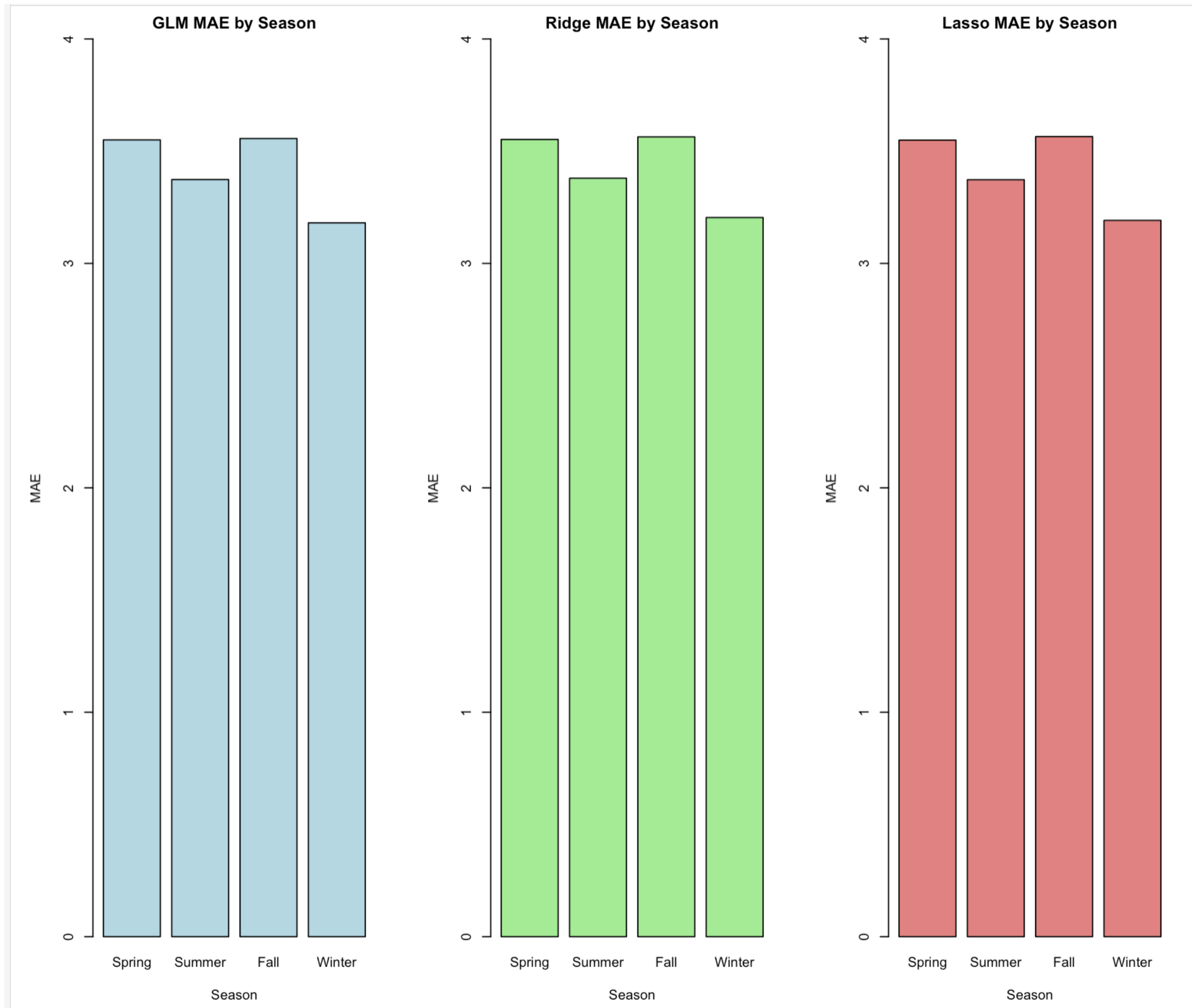


Figure 5: The error from the GLM, Ridge, and Lasso regression models. The vertical bars for each of the models' subplots are the seasons.

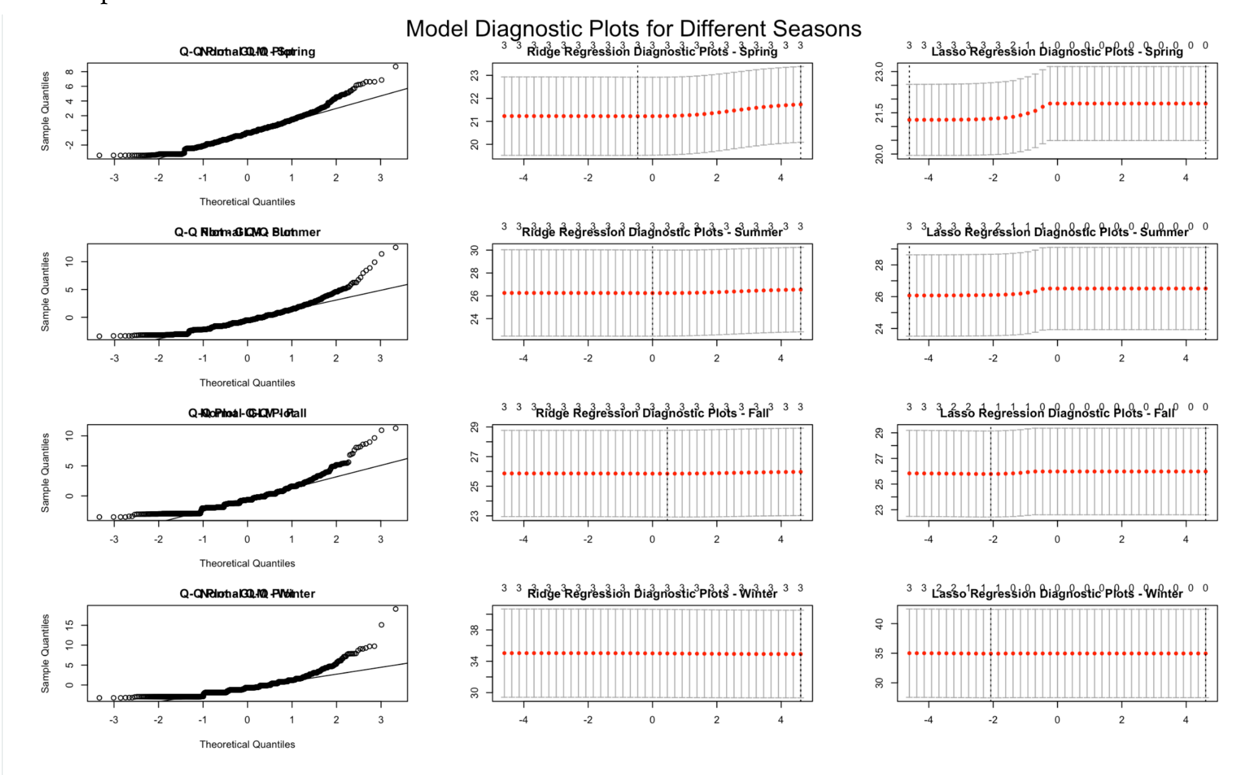
The MAE values for the three models are very similar within each season, indicating that no single model is significantly better than the others across all seasons. In Spring, Lasso Regression shows a slightly lower MAE (3.549097) compared to GLM (3.550265) and Ridge Regression (3.552154). In Summer, Lasso Regression again performs marginally better (3.372767) than GLM (3.373538) and Ridge Regression (3.379585). In Fall, GLM (3.556377) and Ridge Regression (3.563616) have slightly lower MAEs compared to Lasso Regression (3.564943). In Winter, GLM (3.180541) shows the best performance, followed closely by Lasso Regression (3.191702) and Ridge Regression (3.204355).

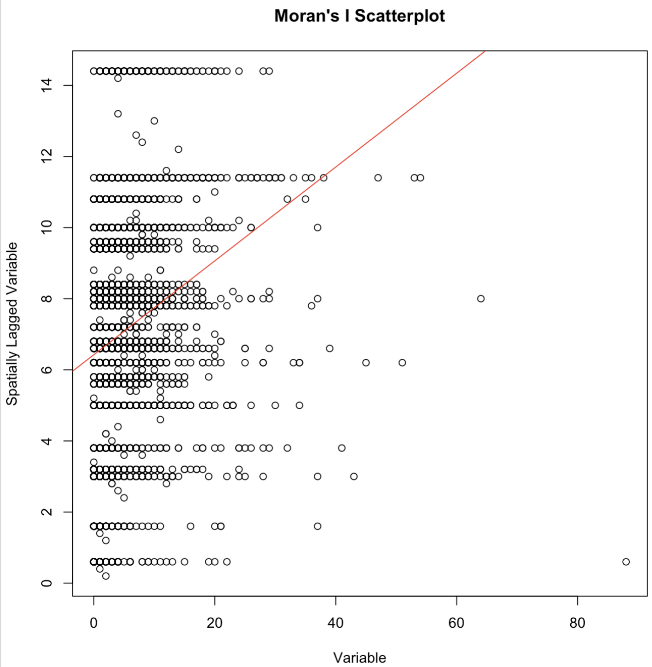
In conclusion, all three regression models exhibit similar performance with only minor differences in MAE across the seasons. Winter appears to be the season with the most accurate predictions, while Fall is the most challenging for these models. Among the models, Lasso Regression generally performs slightly better in Spring and Summer, whereas GLM performs best in Winter.

This analysis indicates that it is possible to predict the abundance of birds in Jasper Ridge with a reasonable degree of accuracy using GLM, Ridge Regression, or Lasso Regression. The models exhibit similar performance across seasons, although slight variations in their predictive accuracy are observed. Winter predictions are the most accurate, while Fall predictions are the most challenging. Leveraging these models can provide valuable insights into bird abundance trends, aiding conservation efforts and ecological studies in Jasper Ridge. Considering seasonal adjustments or model combinations may further enhance predictive performance, ensuring more reliable predictions year-round.

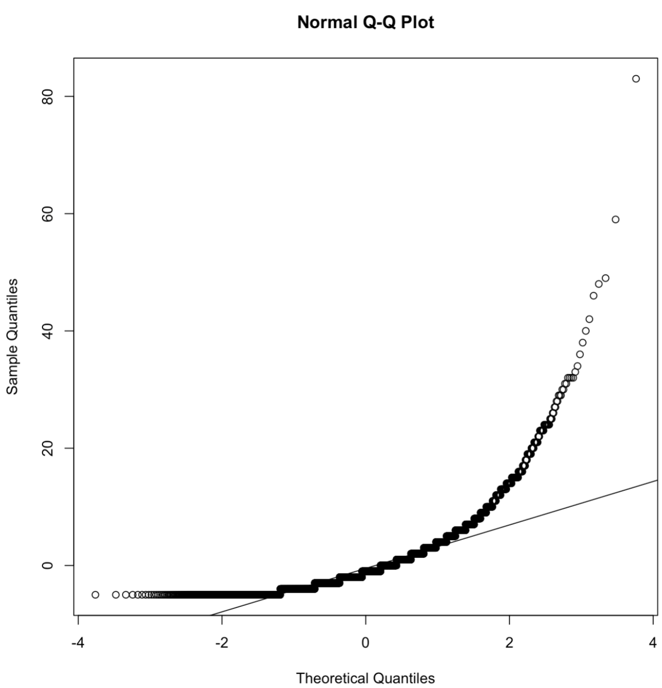
Appendix:

Unused plots:

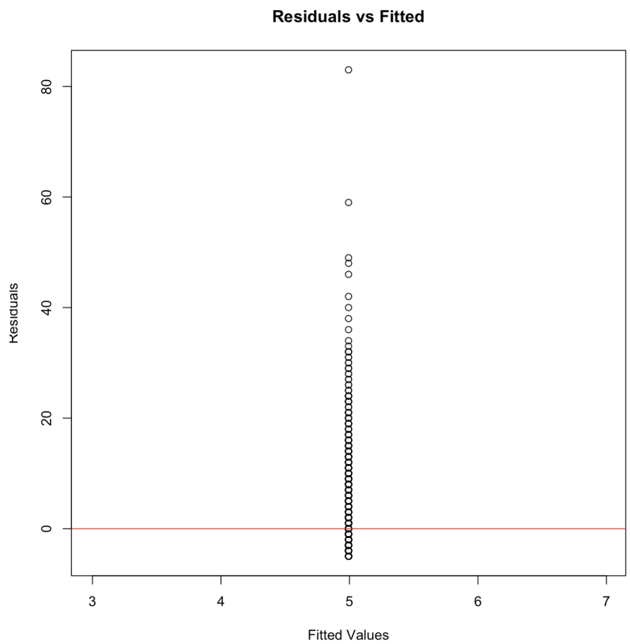




Moran's I Scatterplot
Spatially Lagged Variable
Variable
0
2
4
6
8
10
12
14
0
20
40
60
80



Normal Q-Q Plot
Sample Quantiles
Theoretical Quantiles
0
20
40
60
80
-4
-2
0
2
4



Code:

Load required libraries

library(spdep)

library(nlme)

library(MASS)

library(ggplot2)

library(dplyr)

library(reshape2)

library(sp)

library(raster)

library(glmnet)

```r
# Step 1: Data Preparation

# Load the dataset
# Set working directory
setwd("/Users/pragnyavijayan/Spring 2024/Advanced Analysis of Biological Data/project1")
# Load datasets
bird_data <- read.csv('Jasper ridge birds.csv')

# Sum up the counts of bird sightings to get total bird count
bird_data$n.total <- rowSums(bird_data[,2:128])

# Convert Date to proper date format
bird_data$Date <- as.Date(bird_data$Date)

# Extract month from the date
bird_data$Month <- bird_data$Month

# Extract year from the date
bird_data$Year <- bird_data$Year

# Check for missing values
summary(bird_data)
```

```
# Organize data by season
spring_data <- bird_data[bird_data$Month %in% c("3", "4", "5"), ]
summer_data <- bird_data[bird_data$Month %in% c("6", "7", "8"), ]
fall_data <- bird_data[bird_data$Month %in% c("9", "10", "11"), ]
winter_data <- bird_data[bird_data$Month %in% c("12", "1", "2"), ]

# Step 2: Exploratory Data Analysis (EDA)

# Visualize the distribution of bird sightings
ggplot(bird_data, aes(x = n.total)) +
  geom_histogram(binwidth = 10, fill = "blue", color = "black") +
  labs(title = "Distribution of Bird Sightings",
       x = "Total Bird Count",
       y = "Frequency") +
  theme_minimal()

# Plot the temporal distribution of bird sightings by month
ggplot(bird_data, aes(x = Month)) +
  geom_bar(fill = "green", color = "black") +
  labs(title = "Temporal Distribution of Bird Sightings by Month",
       x = "Month",
       y = "Frequency") +
```

```
  theme_minimal()

# Visualize the land cover changes over time
land_cover_cols <- c("wetland.50", "grassland.50", "shrub.50", "water.50")
land_cover_data <- bird_data[,c("Year", "Month", land_cover_cols)]

# Melt the data for easier plotting
land_cover_data <- melt(bird_data[, c("Year", "Month", land_cover_cols)], id.vars = c("Year",
"Month"), variable.name = "Land_Cover", value.name = "Percentage")

# Plot
ggplot(land_cover_data, aes(x = factor(Year), y = Percentage, color = Land_Cover, group =
interaction(Year, Land_Cover))) +
  geom_point(position = position_dodge(width = 0.5)) +
  geom_line(position = position_dodge(width = 0.5)) +
  labs(title = "Land Cover Changes Over Time by Year",
       x = "Year",
       y = "Percentage of Land Cover",
       color = "Land Cover") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))

last_year <- max(bird_data$Year)
land_cover_data_last_year <- subset(land_cover_data, Year >= last_year)
```

```
# Create separate plots for each land cover type

plots <- lapply(land_cover_cols, function(cover) {

  ggplot(subset(land_cover_data_last_year, Land_Cover == cover), aes(x = Month, y = Percentage,
color = Land_Cover, group = interaction(Year, Land_Cover))) +

    #geom_point(position = position_dodge(width = 0.5)) +

    geom_line(position = position_dodge(width = 0.5)) +

    labs(title = paste("Land Cover Changes Over Time by Month -", cover),

         x = "Month",

         y = "Percentage of Land Cover",

         color = "Land Cover") +

    theme_minimal() +

    theme(axis.text.x = element_text(angle = 45, hjust = 1))

})


# Print the plots

gridExtra::grid.arrange(grobs = plots, ncol = 2)


# Step 3: Spatial Autocorrelation Check


# Convert bird_data to a spatial object

coordinates(bird_data) <- ~coords.x1 + coords.x2


# Create a spatial weights matrix
```

```
bird_data.nb <- knn2nb(knearneigh(coordinates(bird_data), k = 5))
bird_data.listw <- nb2listw(bird_data.nb, style = "W")

# Perform Moran's I test
moran <- moran.test(bird_data$n.total, bird_data.listw)
moran

plot(bird_data$n.total, lag.listw(bird_data.listw, bird_data$n.total),
    xlab = "Variable", ylab = "Spatially Lagged Variable",
    main = "Moran's I Scatterplot")
abline(lm(lag.listw(bird_data.listw, bird_data$n.total) ~ bird_data$n.total), col = "red")
```

'

```
Moran I statistic standard deviate = 22.853, p-value < 2.2e-16
alternative hypothesis: greater
sample estimates:
Moran I statistic      Expectation        Variance
    1.319538e-01    -1.691189e-04     3.342472e-05
```

A strong positive spatial autocorrelation in the total bird count data,

meaning that areas with similar total bird counts tend to be clustered together

spatially.

'

```
# Step 4: Temporal Autocorrelation Check

# Calculate autocorrelation
model <- gls(n.total ~ 1, data = bird_data, correlation = corAR1(form = ~ 1 | Month))
model

# Get residuals and fitted values
residuals <- resid(model)
fitted_values <- fitted(model)

# Create QQ plot
qqnorm(residuals)
qqline(residuals)

# Plot residuals against fitted values
plot(fitted_values, residuals, xlab = "Fitted Values", ylab = "Residuals",
    main = "Residuals vs Fitted")
abline(h = 0, col = "red")
```

'

Generalized least squares fit by REML

Model: n.total ~ 1

Data: bird_data

Log-restricted-likelihood: -18123.83

Coefficients:

(Intercept)

4.992926

Correlation Structure: AR(1)

Formula: ~1 | Month

Parameter estimate(s):

Phi

0.1654994

Degrees of freedom: 5914 total; 5913 residual

Residual standard error: 5.255619

The model suggests that there is a significant relationship between the total

bird count and time (month), taking into account the autocorrelation within each month

'

##

Spring

```
set.seed(123)

train_index_spring <- sample(1:nrow(spring_data), 0.8 * nrow(spring_data))

train_data_spring <- spring_data[train_index_spring, ]

test_data_spring <- spring_data[-train_index_spring, ]


# Summer

set.seed(123)

train_index_summer <- sample(1:nrow(summer_data), 0.8 * nrow(summer_data))

train_data_summer <- summer_data[train_index_summer, ]

test_data_summer <- summer_data[-train_index_summer, ]


# Fall

set.seed(123)

train_index_fall <- sample(1:nrow(fall_data), 0.8 * nrow(fall_data))

train_data_fall <- fall_data[train_index_fall, ]

test_data_fall <- fall_data[-train_index_fall, ]


# Winter

set.seed(123)

train_index_winter <- sample(1:nrow(winter_data), 0.8 * nrow(winter_data))

train_data_winter <- winter_data[train_index_winter, ]

test_data_winter <- winter_data[-train_index_winter, ]
```

```
# Convert train_data to a spatial object

coordinates(train_data_spring) <- ~coords.x1 + coords.x2
coordinates(train_data_summer) <- ~coords.x1 + coords.x2
coordinates(train_data_fall) <- ~coords.x1 + coords.x2
coordinates(train_data_winter) <- ~coords.x1 + coords.x2

# Check for identical points
if (length(unique(coordinates(train_data_spring))) == 1 ||
    length(unique(coordinates(train_data_summer))) == 1 ||
    length(unique(coordinates(train_data_fall))) == 1 ||
    length(unique(coordinates(train_data_winter))) == 1) {
  stop("Identical points found in the train_data. Spatial weights cannot be calculated.")
}

# Create a spatial weights matrix for each season
train_data_matrix_spring.nb <- knn2nb(knearneigh(coordinates(train_data_spring), k = 5))
train_data_matrix_summer.nb <- knn2nb(knearneigh(coordinates(train_data_summer), k = 5))
train_data_matrix_fall.nb <- knn2nb(knearneigh(coordinates(train_data_fall), k = 5))
train_data_matrix_winter.nb <- knn2nb(knearneigh(coordinates(train_data_winter), k = 5))

train_data_matrix_spring.listw <- nb2listw(train_data_matrix_spring.nb, style = "W")
train_data_matrix_summer.listw <- nb2listw(train_data_matrix_summer.nb, style = "W")
```

```
train_data_matrix_fall.listw <- nb2listw(train_data_matrix_fall.nb, style = "W")

train_data_matrix_winter.listw <- nb2listw(train_data_matrix_winter.nb, style = "W")


# Calculate lag variables for each season

train_data_spring$wetland_lag <- lag.listw(train_data_matrix_spring.listw,
train_data_spring[["wetland.50"]])

train_data_spring$grassland_lag <- lag.listw(train_data_matrix_spring.listw,
train_data_spring[["grassland.50"]])

train_data_spring$shrub_lag <- lag.listw(train_data_matrix_spring.listw,
train_data_spring[["shrub.50"]])

train_data_spring$water_lag <- lag.listw(train_data_matrix_spring.listw,
train_data_spring[["water.50"]])


train_data_summer$wetland_lag <- lag.listw(train_data_matrix_summer.listw,
train_data_summer[["wetland.50"]])

train_data_summer$grassland_lag <- lag.listw(train_data_matrix_summer.listw,
train_data_summer[["grassland.50"]])

train_data_summer$shrub_lag <- lag.listw(train_data_matrix_summer.listw,
train_data_summer[["shrub.50"]])

train_data_summer$water_lag <- lag.listw(train_data_matrix_summer.listw,
train_data_summer[["water.50"]])


train_data_fall$wetland_lag <- lag.listw(train_data_matrix_fall.listw, train_data_fall[["wetland.50"]])

train_data_fall$grassland_lag <- lag.listw(train_data_matrix_fall.listw,
train_data_fall[["grassland.50"]])

train_data_fall$shrub_lag <- lag.listw(train_data_matrix_fall.listw, train_data_fall[["shrub.50"]])

train_data_fall$water_lag <- lag.listw(train_data_matrix_fall.listw, train_data_fall[["water.50"]])
```

```
train_data_winter$wetland_lag <- lag.listw(train_data_matrix_winter.listw,
train_data_winter[["wetland.50"]])

train_data_winter$grassland_lag <- lag.listw(train_data_matrix_winter.listw,
train_data_winter[["grassland.50"]])

train_data_winter$shrub_lag <- lag.listw(train_data_matrix_winter.listw,
train_data_winter[["shrub.50"]])

train_data_winter$water_lag <- lag.listw(train_data_matrix_winter.listw,
train_data_winter[["water.50"]])



# Calculate lag variables for each season in test data


coordinates(test_data_spring) <- ~coords.x1 + coords.x2

coordinates(test_data_summer) <- ~coords.x1 + coords.x2

coordinates(test_data_fall) <- ~coords.x1 + coords.x2

coordinates(test_data_winter) <- ~coords.x1 + coords.x2



test_data_matrix_spring.nb <- knn2nb(knearneigh(coordinates(test_data_spring), k = 5))

test_data_matrix_summer.nb <- knn2nb(knearneigh(coordinates(test_data_summer), k = 5))

test_data_matrix_fall.nb <- knn2nb(knearneigh(coordinates(test_data_fall), k = 5))

test_data_matrix_winter.nb <- knn2nb(knearneigh(coordinates(test_data_winter), k = 5))


test_data_matrix_spring.listw <- nb2listw(test_data_matrix_spring.nb, style = "W")
```

```
test_data_matrix_summer.listw <- nb2listw(test_data_matrix_summer.nb, style = "W")

test_data_matrix_fall.listw <- nb2listw(test_data_matrix_fall.nb, style = "W")

test_data_matrix_winter.listw <- nb2listw(test_data_matrix_winter.nb, style = "W")


test_data_spring$wetland_lag <- lag.listw(test_data_matrix_spring.listw,
test_data_spring[["wetland.50"]])

test_data_spring$grassland_lag <- lag.listw(test_data_matrix_spring.listw,
test_data_spring[["grassland.50"]])

test_data_spring$shrub_lag <- lag.listw(test_data_matrix_spring.listw, test_data_spring[["shrub.50"]])

test_data_spring$water_lag <- lag.listw(test_data_matrix_spring.listw, test_data_spring[["water.50"]])


test_data_summer$wetland_lag <- lag.listw(test_data_matrix_summer.listw,
test_data_summer[["wetland.50"]])

test_data_summer$grassland_lag <- lag.listw(test_data_matrix_summer.listw,
test_data_summer[["grassland.50"]])

test_data_summer$shrub_lag <- lag.listw(test_data_matrix_summer.listw,
test_data_summer[["shrub.50"]])

test_data_summer$water_lag <- lag.listw(test_data_matrix_summer.listw,
test_data_summer[["water.50"]])


test_data_fall$wetland_lag <- lag.listw(test_data_matrix_fall.listw, test_data_fall[["wetland.50"]])

test_data_fall$grassland_lag <- lag.listw(test_data_matrix_fall.listw, test_data_fall[["grassland.50"]])

test_data_fall$shrub_lag <- lag.listw(test_data_matrix_fall.listw, test_data_fall[["shrub.50"]])

test_data_fall$water_lag <- lag.listw(test_data_matrix_fall.listw, test_data_fall[["water.50"]])
```

```
test_data_winter$wetland_lag <- lag.listw(test_data_matrix_winter.listw,
test_data_winter[["wetland.50"]])

test_data_winter$grassland_lag <- lag.listw(test_data_matrix_winter.listw,
test_data_winter[["grassland.50"]])

test_data_winter$shrub_lag <- lag.listw(test_data_matrix_winter.listw,
test_data_winter[["shrub.50"]])

test_data_winter$water_lag <- lag.listw(test_data_matrix_winter.listw,
test_data_winter[["water.50"]])



# Fit GLM with Spatial Filters on training data for each season

glm_model_spring <- glm(n.total ~ wetland_lag + grassland_lag + shrub_lag,

            data = train_data_spring,

            family = poisson)


glm_model_summer <- glm(n.total ~ wetland_lag + grassland_lag + shrub_lag,

            data = train_data_summer,

            family = poisson)


glm_model_fall <- glm(n.total ~ wetland_lag + grassland_lag + shrub_lag,

           data = train_data_fall,

           family = poisson)


glm_model_winter <- glm(n.total ~ wetland_lag + grassland_lag + shrub_lag,

            data = train_data_winter,
```

```
            family = poisson)

# Predict on test data for each season
glm_pred_spring <- predict(glm_model_spring, newdata = test_data_spring, type = "response")
glm_pred_summer <- predict(glm_model_summer, newdata = test_data_summer, type = "response")
glm_pred_fall <- predict(glm_model_fall, newdata = test_data_fall, type = "response")
glm_pred_winter <- predict(glm_model_winter, newdata = test_data_winter, type = "response")

# Calculate MAE for GLM for each season
glm_mae_spring <- mean(abs(glm_pred_spring - test_data_spring$n.total))
glm_mae_summer <- mean(abs(glm_pred_summer - test_data_summer$n.total))
glm_mae_fall <- mean(abs(glm_pred_fall - test_data_fall$n.total))
glm_mae_winter <- mean(abs(glm_pred_winter - test_data_winter$n.total))

# Fit Ridge Regression for each season
X_train_spring <- model.matrix(~ wetland_lag + grassland_lag + shrub_lag + water_lag, data =
train_data_spring)
X_train_summer <- model.matrix(~ wetland_lag + grassland_lag + shrub_lag + water_lag, data =
train_data_summer)
X_train_fall <- model.matrix(~ wetland_lag + grassland_lag + shrub_lag + water_lag, data =
train_data_fall)
X_train_winter <- model.matrix(~ wetland_lag + grassland_lag + shrub_lag + water_lag, data =
train_data_winter)

X_test_spring <- model.matrix(~ wetland_lag + grassland_lag + shrub_lag + water_lag, data =
test_data_spring)
```

```
X_test_summer <- model.matrix(~ wetland_lag + grassland_lag + shrub_lag + water_lag, data =
test_data_summer)

X_test_fall <- model.matrix(~ wetland_lag + grassland_lag + shrub_lag + water_lag, data =
test_data_fall)

X_test_winter <- model.matrix(~ wetland_lag + grassland_lag + shrub_lag + water_lag, data =
test_data_winter)


y_train_spring <- train_data_spring$n.total

y_train_summer <- train_data_summer$n.total

y_train_fall <- train_data_fall$n.total

y_train_winter <- train_data_winter$n.total


# Perform cross-validated ridge regression with a different lambda sequence for each season

cv_ridge_model_spring <- cv.glmnet(X_train_spring, y_train_spring, alpha = 0, lambda = 10^seq(-2,
2, 0.1))

cv_ridge_model_summer <- cv.glmnet(X_train_summer, y_train_summer, alpha = 0, lambda =
10^seq(-2, 2, 0.1))

cv_ridge_model_fall <- cv.glmnet(X_train_fall, y_train_fall, alpha = 0, lambda = 10^seq(-2, 2, 0.1))

cv_ridge_model_winter <- cv.glmnet(X_train_winter, y_train_winter, alpha = 0, lambda = 10^seq(-2,
2, 0.1))


# Best lambda value for each season

best_lambda_spring <- cv_ridge_model_spring$lambda.min

best_lambda_summer <- cv_ridge_model_summer$lambda.min

best_lambda_fall <- cv_ridge_model_fall$lambda.min

best_lambda_winter <- cv_ridge_model_winter$lambda.min
```

```
# Predict using the best lambda for each season

ridge_pred_spring <- predict(cv_ridge_model_spring, s = best_lambda_spring, newx = X_test_spring)

ridge_pred_summer <- predict(cv_ridge_model_summer, s = best_lambda_summer, newx =
X_test_summer)

ridge_pred_fall <- predict(cv_ridge_model_fall, s = best_lambda_fall, newx = X_test_fall)

ridge_pred_winter <- predict(cv_ridge_model_winter, s = best_lambda_winter, newx =
X_test_winter)


# Calculate MAE for Ridge Regression for each season

ridge_mae_spring <- mean(abs(ridge_pred_spring - test_data_spring$n.total))

ridge_mae_summer <- mean(abs(ridge_pred_summer - test_data_summer$n.total))

ridge_mae_fall <- mean(abs(ridge_pred_fall - test_data_fall$n.total))

ridge_mae_winter <- mean(abs(ridge_pred_winter - test_data_winter$n.total))


# Fit Lasso Regression for each season

cv_lasso_model_spring <- cv.glmnet(X_train_spring, y_train_spring, alpha = 1, lambda = 10^seq(-2,
2, 0.1))

cv_lasso_model_summer <- cv.glmnet(X_train_summer, y_train_summer, alpha = 1, lambda =
10^seq(-2, 2, 0.1))

cv_lasso_model_fall <- cv.glmnet(X_train_fall, y_train_fall, alpha = 1, lambda = 10^seq(-2, 2, 0.1))

cv_lasso_model_winter <- cv.glmnet(X_train_winter, y_train_winter, alpha = 1, lambda = 10^seq(-2,
2, 0.1))


# Best lambda value for each season

best_lambda_lasso_spring <- cv_lasso_model_spring$lambda.min
```

```
best_lambda_lasso_summer <- cv_lasso_model_summer$lambda.min

best_lambda_lasso_fall <- cv_lasso_model_fall$lambda.min

best_lambda_lasso_winter <- cv_lasso_model_winter$lambda.min


# Predict using the best lambda for each season

lasso_pred_spring <- predict(cv_lasso_model_spring, s = best_lambda_lasso_spring, newx =
X_test_spring)

lasso_pred_summer <- predict(cv_lasso_model_summer, s = best_lambda_lasso_summer, newx =
X_test_summer)

lasso_pred_fall <- predict(cv_lasso_model_fall, s = best_lambda_lasso_fall, newx = X_test_fall)

lasso_pred_winter <- predict(cv_lasso_model_winter, s = best_lambda_lasso_winter, newx =
X_test_winter)


# Calculate MAE for Lasso Regression for each season

lasso_mae_spring <- mean(abs(lasso_pred_spring - test_data_spring$n.total))

lasso_mae_summer <- mean(abs(lasso_pred_summer - test_data_summer$n.total))

lasso_mae_fall <- mean(abs(lasso_pred_fall - test_data_fall$n.total))

lasso_mae_winter <- mean(abs(lasso_pred_winter - test_data_winter$n.total))


# Print MAE for each model and season

cat("GLM MAE for Spring:", glm_mae_spring, "\n")

cat("GLM MAE for Summer:", glm_mae_summer, "\n")

cat("GLM MAE for Fall:", glm_mae_fall, "\n")

cat("GLM MAE for Winter:", glm_mae_winter, "\n")
```

```
cat("Ridge Regression MAE for Spring:", ridge_mae_spring, "\n")
cat("Ridge Regression MAE for Summer:", ridge_mae_summer, "\n")
cat("Ridge Regression MAE for Fall:", ridge_mae_fall, "\n")
cat("Ridge Regression MAE for Winter:", ridge_mae_winter, "\n")

cat("Lasso Regression MAE for Spring:", lasso_mae_spring, "\n")
cat("Lasso Regression MAE for Summer:", lasso_mae_summer, "\n")
cat("Lasso Regression MAE for Fall:", lasso_mae_fall, "\n")
cat("Lasso Regression MAE for Winter:", lasso_mae_winter, "\n")


# Plot the MAE results
par(mfrow = c(1, 3), mar = c(5, 5, 2, 2))

# MAE for GLM, Ridge, and Lasso by season
barplot(c(glm_mae_spring, glm_mae_summer, glm_mae_fall, glm_mae_winter),
    col = "lightblue", main = "GLM MAE by Season", ylim = c(0, 4),
    names.arg = c("Spring", "Summer", "Fall", "Winter"), ylab = "MAE", xlab = "Season")
barplot(c(ridge_mae_spring, ridge_mae_summer, ridge_mae_fall, ridge_mae_winter),
    col = "lightgreen", main = "Ridge MAE by Season", ylim = c(0, 4),
    names.arg = c("Spring", "Summer", "Fall", "Winter"), ylab = "MAE", xlab = "Season")
barplot(c(lasso_mae_spring, lasso_mae_summer, lasso_mae_fall, lasso_mae_winter),
    col = "lightcoral", main = "Lasso MAE by Season", ylim = c(0, 4),
```

```r
      names.arg = c("Spring", "Summer", "Fall", "Winter"), ylab = "MAE", xlab = "Season")

# Plot the MAE results
par(mfrow = c(1, 3), mar = c(5, 5, 2, 2))

# MAE for GLM, Ridge, and Lasso by model
barplot(c(glm_mae_spring, glm_mae_summer, glm_mae_fall, glm_mae_winter),
      col = "lightblue", main = "GLM MAE by Season", ylim = c(0, 4),
      names.arg = c("Spring", "Summer", "Fall", "Winter"), ylab = "MAE", xlab = "Season")
barplot(c(ridge_mae_spring, ridge_mae_summer, ridge_mae_fall, ridge_mae_winter),
      col = "lightgreen", main = "Ridge MAE by Season", ylim = c(0, 4),
      names.arg = c("Spring", "Summer", "Fall", "Winter"), ylab = "MAE", xlab = "Season")
barplot(c(lasso_mae_spring, lasso_mae_summer, lasso_mae_fall, lasso_mae_winter),
      col = "lightcoral", main = "Lasso MAE by Season", ylim = c(0, 4),
      names.arg = c("Spring", "Summer", "Fall", "Winter"), ylab = "MAE", xlab = "Season")

legend("topright", legend = c("GLM", "Ridge", "Lasso"),
     fill = c("lightblue", "lightgreen", "lightcoral"), bty = "n")




par(mfrow = c(4, 3), oma = c(2, 2, 2, 2))
```

```
# Diagnostic plots for Spring

# GLM model - Spring (Q-Q plot)
qqnorm(residuals(glm_model_spring))
qqline(residuals(glm_model_spring))
title(main = "Q-Q Plot - GLM - Spring")

# Ridge Regression model - Spring (Log(lambda) plot)
plot(cv_ridge_model_spring, main = "Ridge Regression Diagnostic Plots - Spring", xlab = "", ylab = "")
mtext("Log(lambda)", side = 1, line = 2, outer = TRUE)

# Lasso Regression model - Spring (Log(lambda) plot)
plot(cv_lasso_model_spring, main = "Lasso Regression Diagnostic Plots - Spring", xlab = "", ylab = "")
mtext("Log(lambda)", side = 1, line = 2, outer = TRUE)

# Diagnostic plots for Summer

# GLM model - Summer (Q-Q plot)
qqnorm(residuals(glm_model_summer))
qqline(residuals(glm_model_summer))
title(main = "Q-Q Plot - GLM - Summer")

# Ridge Regression model - Summer (Log(lambda) plot)
```

```
plot(cv_ridge_model_summer, main = "Ridge Regression Diagnostic Plots - Summer", xlab = "", ylab = "")

mtext("Log(lambda)", side = 1, line = 2, outer = TRUE)


# Lasso Regression model - Summer (Log(lambda) plot)

plot(cv_lasso_model_summer, main = "Lasso Regression Diagnostic Plots - Summer", xlab = "", ylab = "")

mtext("Log(lambda)", side = 1, line = 2, outer = TRUE)


# Diagnostic plots for Fall


# GLM model - Fall (Q-Q plot)

qqnorm(residuals(glm_model_fall))

qqline(residuals(glm_model_fall))

title(main = "Q-Q Plot - GLM - Fall")


# Ridge Regression model - Fall (Log(lambda) plot)

plot(cv_ridge_model_fall, main = "Ridge Regression Diagnostic Plots - Fall", xlab = "", ylab = "")

mtext("Log(lambda)", side = 1, line = 2, outer = TRUE)


# Lasso Regression model - Fall (Log(lambda) plot)

plot(cv_lasso_model_fall, main = "Lasso Regression Diagnostic Plots - Fall", xlab = "", ylab = "")

mtext("Log(lambda)", side = 1, line = 2, outer = TRUE)
```

```r
# Diagnostic plots for Winter

# GLM model - Winter (Q-Q plot)
qqnorm(residuals(glm_model_winter))
qqline(residuals(glm_model_winter))
title(main = "Q-Q Plot - GLM - Winter")

# Ridge Regression model - Winter (Log(lambda) plot)
plot(cv_ridge_model_winter, main = "Ridge Regression Diagnostic Plots - Winter", xlab = "", ylab = "")
mtext("Log(lambda)", side = 1, line = 2, outer = TRUE)

# Lasso Regression model - Winter (Log(lambda) plot)
plot(cv_lasso_model_winter, main = "Lasso Regression Diagnostic Plots - Winter", xlab = "", ylab = "")
mtext("Log(lambda)", side = 1, line = 2, outer = TRUE)

mtext("Model Diagnostic Plots for Different Seasons", side = 3, outer = TRUE, cex = 1.5)
```