

U.S. Trust Responsibility in the Context of FEMA Disaster Relief to Tribal Nations

I. INTRODUCTION

In 1942, a Supreme Court case between the Seminole Nation and the United States made the ruling that the U.S. must charge itself with “moral obligations of the highest responsibility and trust” in its dealing with indigenous tribes.⁸ This finding is referred to as the “Trust Responsibility” and represents the U.S.’ legal obligation to protect indigenous nations. While the Trust Responsibility has tragic roots in U.S. colonization and dismissive attitudes towards native populations, it has persisted from the early 19th century to this day, and is intended to ensure the protection of tribal lands, assets, and resources.¹⁰

However, the United States has a dubious history when it comes to tribal reparations. The U.S. only created a streamlined process for Tribal Nations to request Federal Emergency Management Aid (FEMA) aid in 2013.¹¹ FEMA aid is support provided by the government to help meet “immediate needs after a disaster, such as shelter and medical expenses.”¹² In light of increasingly frequent and intense natural disasters, the goal of our research is to ensure that the U.S. is being held to the standards it has set for itself. Specifically, we hope to address a gap in the literature by analyzing whether Tribal Nations receive FEMA aid at similar rates to U.S. states.

To this end, we will be using raw data from FEMA’s National Emergency Management Information System regarding major disaster or emergency declaration requests that were either accepted or denied. While acceptances and denials were contained in two separate datasets, they both contain useful features such as: (1) the physical location of the request; (2) whether the requester was a Tribal Nation; (3) the incident type; and (4) the date of declaration. Therefore, we will be able to take a closer look at the impact that being a Tribal Nation can have on whether a request for FEMA aid is accepted.

The first step is exploratory data analysis to understand any imbalances in the data. For example, noting that requests from Tribal Nations are relatively infrequent, and that in general there are far more aid request acceptances than denials. To further analyze the data, we constructed a binary logistic regression model to understand what features are important in determining FEMA aid request results. While the logistic regression model is better in the sense that it is more relevant for policy applications, we also created a random forest model to get a sense of how accurately we can predict aid request results.

II. LITERATURE REVIEW

With the abundance of literature on inequality experienced by indigenous communities, from health¹⁴ to economic conditions⁷, it is not a surprise that there are instances of inequality regarding their receipt of FEMA aid. When the Rodeo-Chediski fire of 2002 tore through the Fort Apache Indian Reservation in Arizona, FEMA promised \$20 million of aid to the affected Arizona counties and Apache lands.⁶ However, while Native tribes are able to receive federal disaster relief, they cannot declare a state of emergency themselves given their sovereignty status. Therefore, tribal nations are not able to secure FEMA aid without a state or federal agency serving as the middleman. Before determining whether a federally recognized disaster should be declared (and aid thus given), FEMA conducts a preliminary damage assessment, in which “an Indian tribe or authorized tribal organization” joins “if possible”, meaning that tribal input is included in aid evaluation only when FEMA deems it convenient.⁶ Further, states’ initial damage assessments are not required to include information about damage for tribal communities.⁹

FEMA’s process for financing disaster relief funds and resource management “...can expose the deep rifts of mistrust inherent in tribal sovereignty”.⁹ The exceedingly slow process of aid receipt tests the

patience of any entity that applies for disaster relief. Add to that the traumatic history of erasure and exploitation experienced by Native peoples at the hands of the U.S. Government, and the federal aid process becomes increasingly complex. For impoverished tribes, delays in the delivery of aid can be viewed as yet another instance of abuse and abandonment by government actors.⁹ For wealthier tribes, state governments may have emotional barriers to providing additional financial assistance to tribes.⁹ States' initial damage assessments are not required to include information about damage to tribal lands and properties and related expenditures.⁹ This creates a situation in which tribal participation is seldom included in preliminary FEMA assessments.⁹

Tribal Nations were granted sovereignty in the eyes of FEMA with the Sandy Recovery Improvement Act of 2013, which amended the Stafford Act to allow tribes direct access to federal disaster aid.¹ While well-intentioned, this policy was not without its flaws. While it gave tribes the same sovereign rights as states when seeking disaster relief, it left the state-level criteria and protocols intact.⁵ For example, the threshold for a state to make a disaster request is a million dollar state impact and a statewide per capita impact of \$1.37 (adjusted annually).⁵ No adjustment has been clarified for tribal impact; it is unclear if this impact should be measured by size of tribal or reservation population, or on the value of the dollar in communities that live many degrees below the poverty line.¹¹ Ultimately, the Sandy Amendment has provided additional opportunities for self-governance that many tribes don't have the ability to meet.¹¹ In 2017, the U.S. Senate determined that FEMA should re-examine its criteria for evaluating disaster declarations for tribal communities, citing the lack of criteria for tribal disaster impact.¹³ Our research aims to investigate how well the Sandy Recovery Improvement Act has been implemented over the past decade and provide clarity around the likelihood of a Tribal Nation being denied FEMA disaster relief.

III. METHODS

A. DATA CLEANING

The two FEMA datasets used for this analysis, 'DisasterDeclarationsSummaries', and 'DeclarationDenials', were loaded into RStudio as 'acceptances' and 'denials', respectively. Rows with null values were not immediately dropped since values in other columns in those rows may become applicable, while the columns with null values may not need to be included in our analysis. First, column names in the 'denials' dataset were manually inspected and adjusted so that the two datasets could be merged. This involved removing an extraneous 'state' column, and renaming the following columns for parity with the 'acceptances' dataset: 'stateAbbreviation' to 'state', 'declarationRequestDate' to 'declarationDate', 'declarationRequestType' to 'declarationType', 'incidentName' to 'declarationTitle', 'requestedIncidentTypes' to 'incidentType', and 'requestedIncidentEndDate' to 'incidentEndDate'. These adjustments were made in accordance with FEMA's metadata for the two datasets, from which we were able to determine that while the column titles may be different between the datasets, they captured the same information.² The two datasets were then merged on the following columns: 'state', 'tribalRequest', 'declarationDate', 'declarationTitle', 'incidentType', 'incidentEndDate', 'incidentId', 'incidentBeginDate', 'id', 'region', and 'declarationType'.

A column that designates from which dataset the data came, 'requestResult', was added to provide a method for encoding our model results. This column encoded the data as 'Denial' or 'Acceptance', based on if the data came from the 'DeclarationDenials' dataset ('Denial') or the DisasterDeclarationSummaries dataset ('Acceptance'). Additionally, 'declarationMonth', extracted the month from the 'declarationDate' column to specify the month in which the aid declaration occurred. The

same process was done to create a 'declarationYear' column, which extracted the year from the 'declarationDate' column. Because Tribal Nations were not permitted to submit their own FEMA aid requests until January 2013¹, all data from 2012 and prior was dropped for our analysis.

B. EXPLORATORY DATA ANALYSIS

Initial exploratory data analysis revealed some important trends. For instance, FEMA has recorded far more acceptances than denials, with 24,303 acceptances and only 136 denials in the last decade (Figure 1). Requests are made for 24 distinct incident types (Figure 2). Similarly to the distribution of acceptances vs. denials, non-tribal (state) requests constitute 24,263 requests post-2012, while tribal requests constitute only 176 requests (Figure 3). Part of this divide is likely due to Tribal Nations not being able to apply for their own declaration requests until 2013. While all pre-2013 data was removed, it is likely that this process is still difficult to navigate given its novelty. Visualizing the number of FEMA aid acceptances over time to Tribal Nations shows a general upward trajectory, with a sharp spike in 2020, likely due to the COVID-19 pandemic (Figure 4.1). Examining the scaled denial rate by incident type for tribal vs. non-tribal requests, tribal requests are denied at a higher rate than non-tribal requests for all incident types for which both types of requests have been made (Figure 8.1). This exploratory data analysis informed our models, detailed below.

C. BINARY LOGISTIC REGRESSION

Our first model was a logistic regression model predicting the probability that a request for aid would be accepted. We started by initializing the model with all potential predictors (whether it was a tribal request, incident type, state, and region). To better understand how our logistic regression model's performance would change as we changed the number of variables included, we also plotted the validation curve using leave-one-out cross-validation (LOOCV). This method was used to identify if our model would overfit or underfit with various numbers of variables. In addition, LOOCV allowed the model to be trained more robustly given the small dataset.

LOOCV was completed for versions of the logistic regression model containing one, two, three, and four variables. While this type of cross validation is computationally expensive, given our small dataset, computational cost was not an issue. The model containing one variable included the tribal request parameter. The model containing two variables included the tribal request and state parameters. The model containing three variables included the tribal request, state, and region parameters. The model containing all variables included the tribal request, state, region, and incident type parameters.

The testing accuracy and average training accuracy were calculated for each version of the model. The testing accuracy was found by using the model to make predictions on 30% of the overall data that was originally held out by the train test split. The average training accuracy was found by taking the average of the cross validation accuracies found through LOOCV, which used 70% of the overall data from our initial train test split.

D. RANDOM FOREST

Given that the binary logistic regression model accuracy was fairly low, a random forest model was employed to attempt to better learn how to classify FEMA requests into acceptances or denials. The variables of interest for this model were 'requestResult' ('Denial' or 'Acceptance'), 'tribalRequest' (whether the FEMA aid request was made by a Tribal Nation), 'incidentType', and state. These variables were numerically encoded for modeling. Since there were far more acceptances than denials in the

combined dataset (24,303 acceptances vs. 136 denials post-2012), the dataset was balanced so that 136 acceptances and 136 denials were included in the model. The data was then split into training (70%) and test (30%) sets. The random forest model was then trained with the formula:

```
Call:
randomForest(formula = requestResult_binary ~ ., data = train_data, importance = TRUE, ntree = 500)
Type of random forest: classification
Number of trees: 500
No. of variables tried at each split: 2
```

This formula assigns the variable `rf_model` with a random forest model that takes `requestResult_binary` (0 = aid denial, 1 = aid acceptance) as the dependent (target) variable, and includes all other columns in the `train_data` dataframe are used as predictors (independent variables). `Importance = TRUE` enables the calculation of variable importance, which measures how much each predictor contributes to the model's accuracy or reduction in impurity. This random forest model includes 500 trees, initially, to seek a balance between accuracy and efficiency.

The model accuracy, feature importance (mean decrease Gini Impurity), and OOB error rate across trees were then determined (Figures 18, 19). The out-of-bag error rate graph shows that the error decreases from 0 to 100 trees, then increases around 200 to 350 trees, then decreases again for 400 to 500 trees (Figure 19). This indicates that the optimal number of trees to limit the out-of-bag error rate is around 100-125 or 500. To limit redundancy and increase efficiency, it would be best to use the smaller number of trees. From this, it was determined that the optimal number of trees to reduce OOB error is approximately 125 trees, which led us to change our random forest formula to the following:

```
Call:
randomForest(formula = requestResult_binary ~ ., data = train_data, importance = TRUE, ntree = 125)
Type of random forest: classification
Number of trees: 125
No. of variables tried at each split: 2
```

This formula was then used to predict on the test data, from which a confusion matrix and summary statistics were generated.

IV. RESULTS

The literature tells a clear story about the inequity that Tribal Nations have faced in their dealings with the U.S. government, suggesting that Tribal status may have a negative impact on their receiving federal emergency aid. To uncover what characteristics lead to a disaster declaration being accepted or denied, and to understand if Tribal Nation status is a key contributor to denial, we evaluated the performance and most important features in our two machine learning models.

A. BINARY LOGISTIC REGRESSION

The original regression model using all predictors had low predictive power, with an accuracy of 59%. However, it did find that the only statistically significant feature in our model was whether a request came from a tribal nation. The corresponding feature had a negative coefficient with a p-value of 0.00887.

Through performing leave-one-out cross validation on the four versions of the model with various numbers of variables included, we found that the training and testing accuracies did not have significant variation. The average training accuracy found for each of the four models ranged from 0.57 to 0.60, and the testing accuracy found for each ranged from 0.55 to 0.59. In the validation curve, the curve shows that the training and testing accuracies are closest in value for the model with one parameter and the model

with four parameters (Figure 12). The best model, based on both the average training and testing accuracies, was found to be the model with all four variables included.

The confusion matrix for the final logistic regression model shows that 47 of the samples in the testing dataset were classified correctly, while 33 were classified incorrectly (Figure 13). The overall accuracy was found to be 0.59 (95% CI 0.4718, 0.6965). The precision was 0.60 and the recall was 0.52. The F1 score was 0.56 (Figure 14).

The model summary shows that the only variable with a statistically significant p-value was the tribal request variable (tribalRequest) (Figure 15). The p-value for the tribal request variable was 0.0132. The p-value for the state variable was 0.0660, which was lower than the other two variables, but not low enough to be considered statistically significant based upon its value being less than 0.05. The p-values of the incident type and region variables were 0.3963 and 0.3621, respectively. Taking a look at the coefficients, we can see that the coefficient value for the tribal request variable is -2.64, showing that it has a negative correlation to the final classification of the model.

B. RANDOM FOREST

From the initial random forest model trained with 500 trees, the accuracy was determined to be 0.7 (95% CI 0.5872, 0.7974) after applying the model to the test data (Figure 17). This model generated a p-value of 0.00023 (Figure 17). However, plotting the Out-of-Bag (OOB) Error Rate across trees revealed that the optimum number of trees to reduce OOB error is around 125 trees (Figure 19). From the random forest model trained with 125 trees, the accuracy was 0.7125 (95% CI 0.6005, 0.8082) after applying the model to the test data (Figure 21). The low p-value this model generated ($9.156e-05$) indicates that this model's accuracy is significantly better than random guessing (Figure 21). The accuracy of the tuned random forest model is a slight increase from that of the random forest model with 500 trees. Additionally, the p-value of the tuned random forest model is lower than that of the 500-tree model.

The incident type was found to be the most important feature to split on within the random forest decision trees, followed by the state, region, and then if the request was a tribal request (Figure 18). The importance of the incident type compared to if the request was a tribal request is roughly 45 compared to 5, respectively.

V. DISCUSSION

A. BINARY LOGISTIC REGRESSION

Overall, the final logistic regression model including the tribal request, state, incident type, and region parameters classified the testing data into accepted and denied requests with a 59% accuracy rate. This model accuracy is very low, considering that there are only two classes that the model classifies into. In looking at the validation curve, we saw that each of the four iterations of the model with increasing numbers of parameters included all performed similarly (Figure 12). This result indicates that the features available to us in the dataset do not allow our logistic regression model to learn how to classify FEMA requests into acceptances and denials.

Furthermore, the other performance metrics calculated for our final model show that its performance is not ideal. The precision score shows that only 60% of the samples the model classified as accepted requests were true accepted requests. In addition, the recall score shows that only 52% of the samples that were true accepted requests were correctly identified as accepted requests.

The model summary showed us that only one variable was statistically significant with a p-value less than 0.05 (Figure 15). The tribal request parameter had a p-value of 0.0132 and a negative coefficient

of -2.64. This results means that in this logistic regression model, if a request was made by a Tribal Nation, the request is more likely to be denied. However, it is important to note that due to the low accuracy of our model, the significance of the tribal request parameter is not a strong conclusion.

B. RANDOM FOREST

This model reported better accuracy with a score of 0.7. The incident type was found to be the best feature to split on within the random forest decision trees, followed by the state, region, and then tribal status (Figure 18). The importance of the incident type compared to if the request was a tribal request is roughly 45 compared to 5, respectively. This indicates that whether or not a request was from a Tribal Nation was not found to be as clear of a determining factor for acceptance or denial as the incident type that led to the aid request.

The out-of-bag error rate graph shows that the error decreases from 0 to 100 trees, then increases around 200 to 350 trees, then decreases again for 400 to 500 trees (Figure 19). This indicates that the optimal number of trees is around 100-125 or 500. To limit redundancy and increase efficiency, the model was rebuilt with 100 trees, and achieved an accuracy of 0.7. This result shows that the reduction in trees did not negatively affect the correct classification of the testing data, allowing us to reduce unnecessary complexity. The feature importances found for this new model were very similar to the initial model's, showing consistency in the results.

VI. CONCLUSION

As discussed earlier, the literature on the relationship between Tribal Nations and FEMA reflects a history of inequality and distrust, suggesting that tribal nations may be less likely to receive aid. However, our random forest model ranked Tribal Nation status with the lowest feature importance. This is because features with the highest importance are features that most cleanly split the data. Our random forest model found that the incident type was used to split the data into the most pure groups. During our exploratory data analysis, we found that there were a large amount of biological disaster declarations - most likely due to COVID-19 (Figure 2). In addition, we saw the spike in 2020 when looking at the FEMA aid acceptances to Tribal Nations over time, which corresponds with the height of the COVID-19 pandemic (Figure 4.1). One reason for the model highly valuing incident type may be due to these unique, widespread disasters that immediately received aid. Therefore, the inherent structure of random forests may have led to the true importance of the Tribal Nation status feature being hidden.

While our logistic regression model is not accurate enough for us to gauge the extent to which being a tribal nation negatively impacts the probability of acceptance, that the tribal status parameter was statistically significant is notable. The model's structure takes a holistic combination of factors into account at once, whereas random forest makes decisions based on one factor at a time. This makes it more relevant in determining feature importance, which is the goal of our policy-oriented research. Future logistic regression models could incorporate additional parameters such as the amount of aid requested. If such a model were able to achieve a higher accuracy, the resulting significance and correlation assigned to the Tribal Nation status parameter would be able to be accepted with more confidence.

REFERENCES

1. Anonymous. (2013, February 7). Tribes now can apply for emergency declarations. Wind River News, p. 3.
2. Federal Emergency Management Agency. (n.d.). Declaration denials - v1. FEMA OpenFEMA Data. Retrieved December 3, 2024, from <https://www.fema.gov/openfema-data-page/declaration-denials-v1>
3. Federal Emergency Management Agency. (n.d.). Disaster declarations summaries - v2. FEMA OpenFEMA Data. Retrieved December 3, 2024, from <https://www.fema.gov/openfema-data-page/disaster-declarations-summaries-v2>
4. Kim PJ. Social Determinants of Health Inequities in Indigenous Canadians Through a Life Course Approach to Colonialism and the Residential School System. Health Equity. July 25, 2019; 3(1): 378-381. <https://pmc.ncbi.nlm.nih.gov/articles/PMC6657289/>
5. Luft, R. E. (2016). Governing disaster: The politics of tribal sovereignty in the context of (un)natural disaster. Ethnic and Racial Studies, 39(5), 802–820. <https://doi.org/10.1080/01419870.2015.1080376>
6. Pechuli, G. K. (2003, February 26). Apache still at odds with FEMA policies. Knight Ridder Tribune Business News, 1. Retrieved from <https://www.proquest.com/docview/461962646>
7. Randall Akee. Economic Outcomes for Indigenous Peoples in the United States Following the Great Recession. Focus on poverty. March 2022; 38(1); 19-26. <https://www.irp.wisc.edu/resource/economic-outcomes-for-indigenous-peoples-in-the-united-states-following-the-great-recession/>
8. Seminole Nation v. United States, 316 U.S. 286 (1942). <https://supreme.justia.com/cases/federal/us/316/286/>
9. Steinfeld, C. (2012). A social network perspective on coordination in disaster management. Proceedings of the 9th International Conference on Information Systems for Crisis Response and Management (ISCRAM). Retrieved from https://idl.iscrum.org/files/steinfeld/2012/210_Steinfeld2012.pdf
10. United States v. Kagama, 118 U.S. 375 (1886). <https://supreme.justia.com/cases/federal/us/118/375/>
11. U.S. Federal Emergency Management Agency. Tribal Declarations Pilot Guidance. <https://www.fema.gov/disaster/tribal-declarations>
12. U.S. Government Accountability Office. Federal Disaster Assistance: Individual Assistance Requests Often Granted, but FEMA Could Better Document Factors Considered. May 2018. <https://www.gao.gov/assets/gao-18-366.pdf>
13. U.S. Senate, Committee on Indian Affairs. (2017). Emergency management in Indian Country: Improving FEMA's federal-tribal relationship with Indian tribes: Hearing before the Committee on Indian Affairs, United States Senate, One Hundred Fifteenth Congress, first session, February 8, 2017. U.S. Government Publishing Office. Retrieved from <https://www.govinfo.gov/content/pkg/CHRG-115shrg26164/pdf/CHRG-115shrg26164.pdf>

APPENDIX

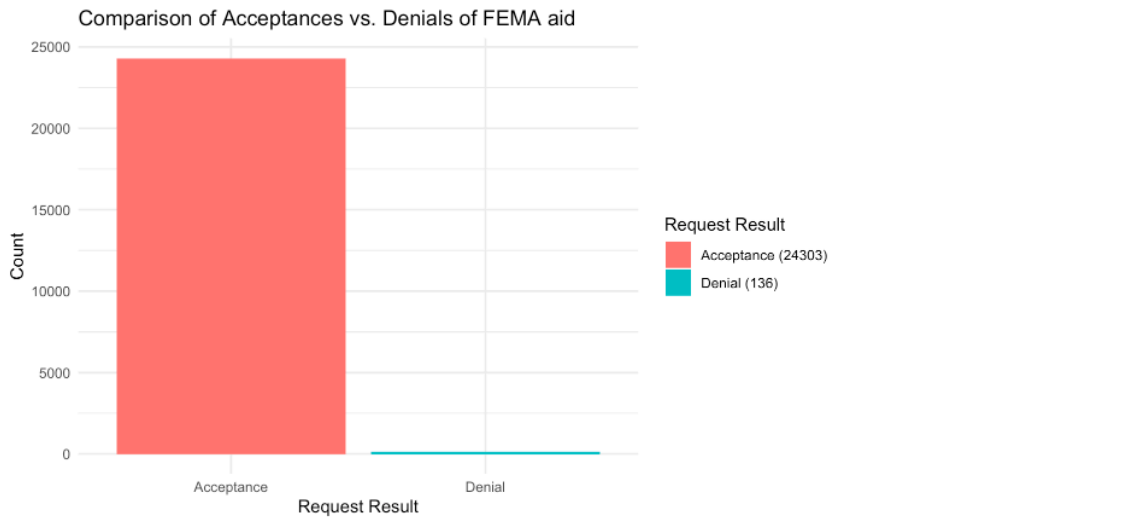


Figure 1. Comparing Overall Acceptances vs. Denials of FEMA aid, we see there are far more documented acceptances than denials of FEMA aid.

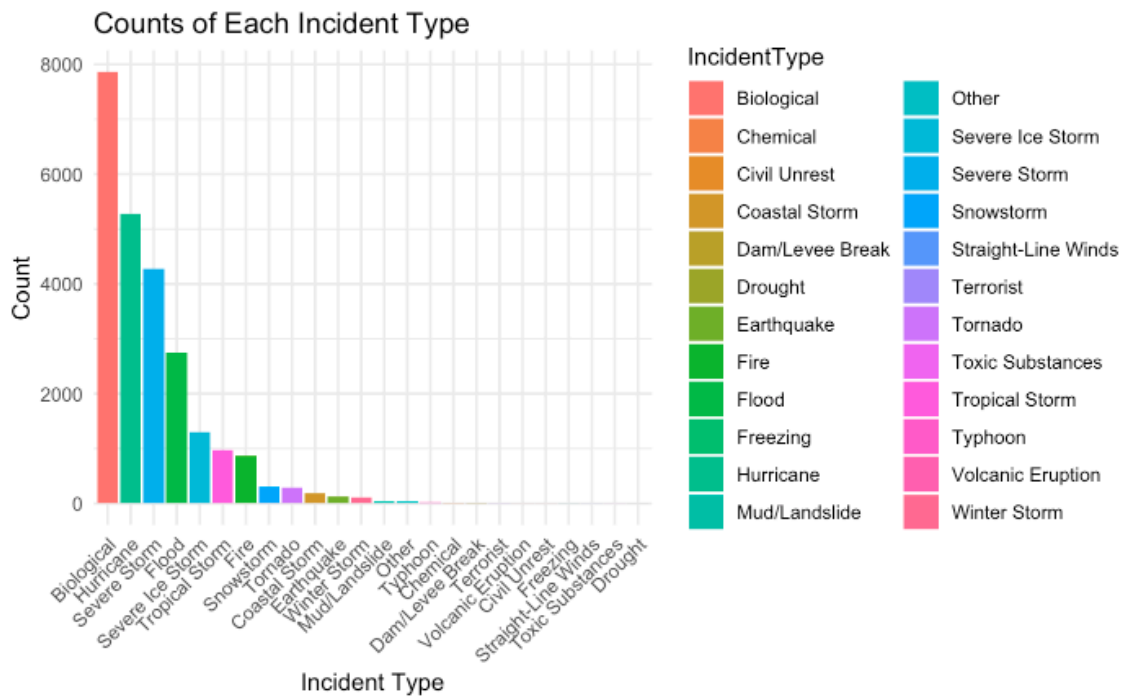


Figure 2. Visualizing the variation among incident types allows us to get a better sense of the various reasons for FEMA aid requests. Biological is the most prevalent incident type, followed by Hurricane, Severe Storm, and Flood.

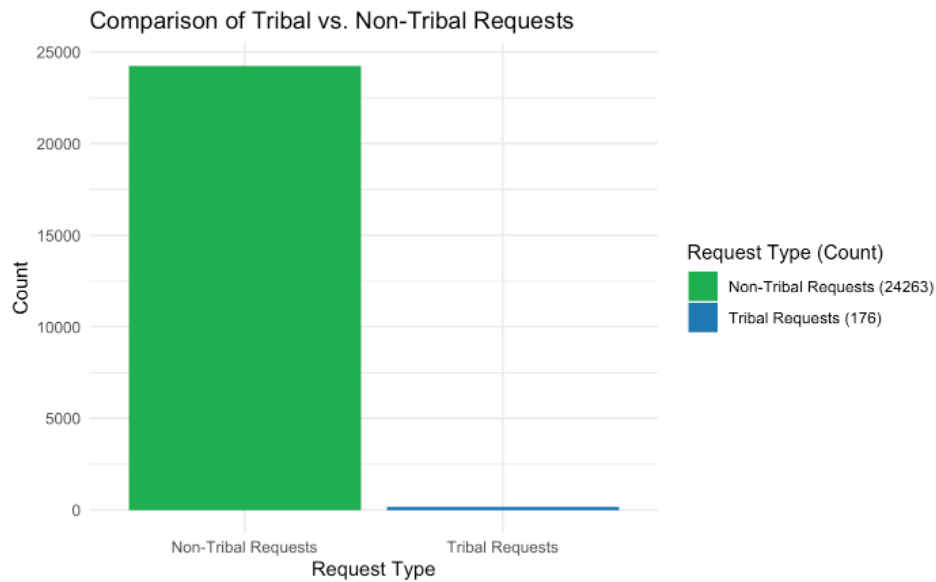


Figure 3. Examining the number of tribal nation vs. non-tribal nation requests shows how infrequent tribal requests are. Part of this divide is likely due to tribal nations not being able to apply for their own declaration request until January 29, 2013. While all pre-2013 data was removed, it is likely that this process is still not as streamlined as the request process for states.

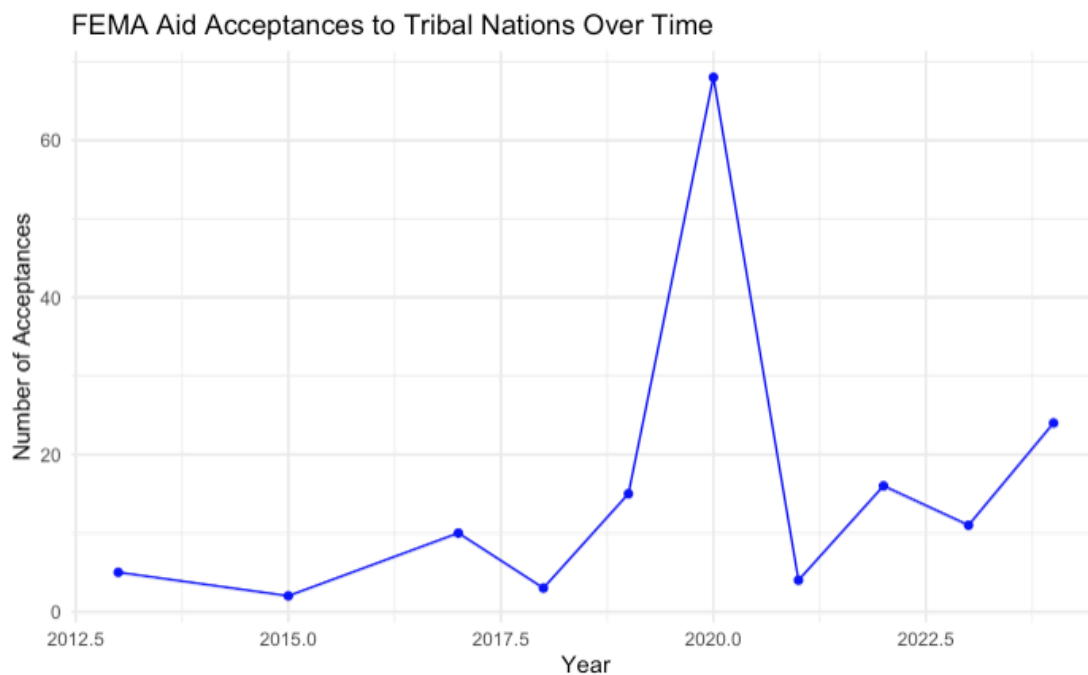


Figure 4.1

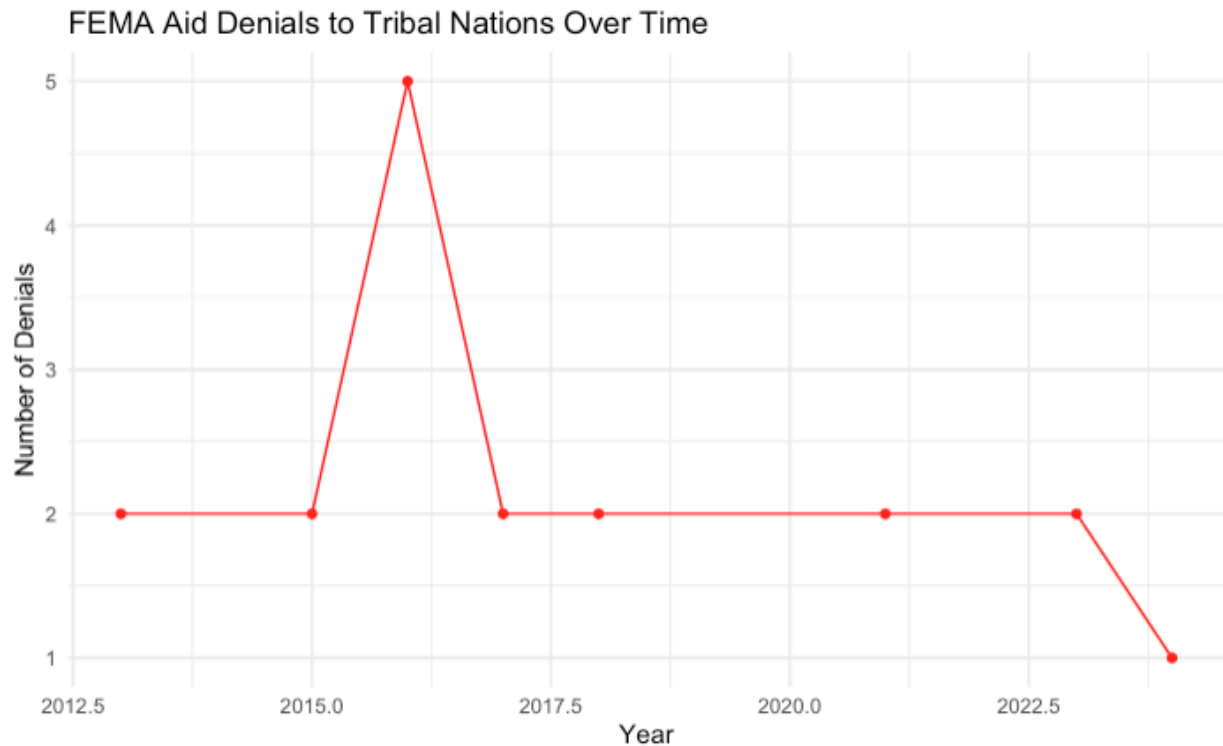


Figure 4.2. Visualizing the number of FEMA aid acceptances over time to tribal nations shows a general upward trajectory, with a sharp spike in 2020, likely due to the COVID-19 pandemic. There appear to be far fewer denials than acceptances.

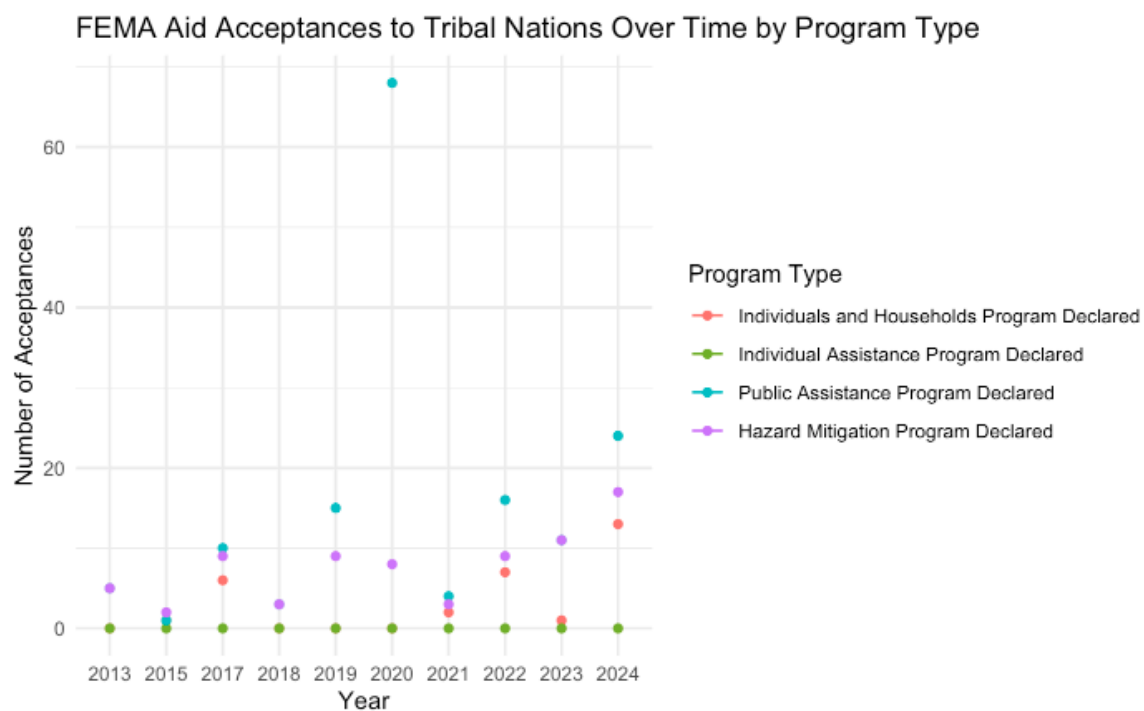


Figure 5.1

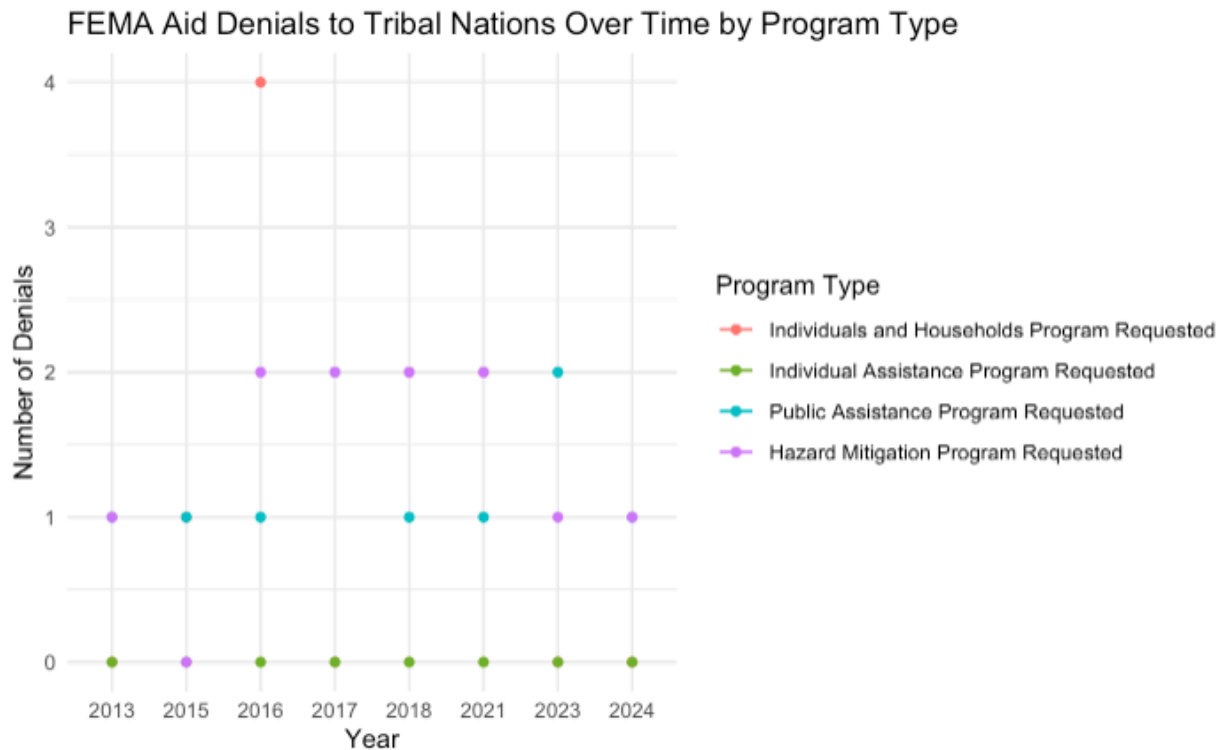


Figure 5.2. Examining the number of acceptances and denials to tribal nations over time, segmented by program type, allows for a broader understanding of what types of aid are being requested.

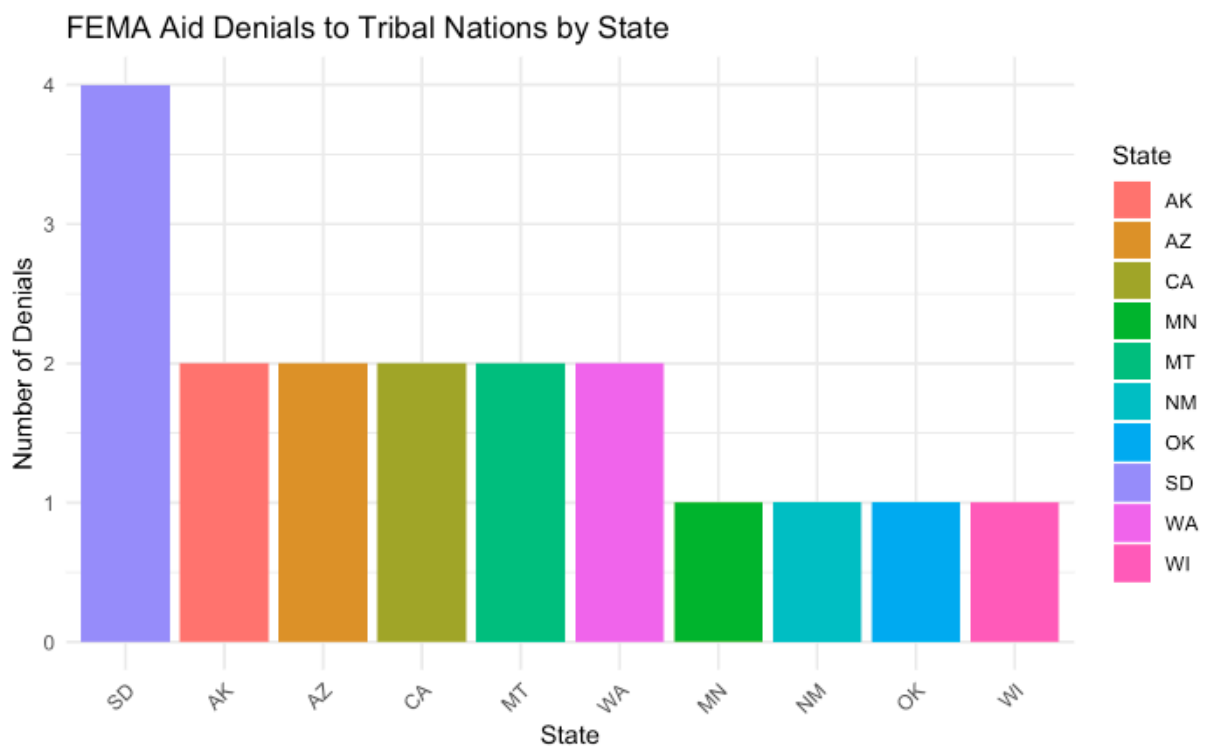


Figure 6.1

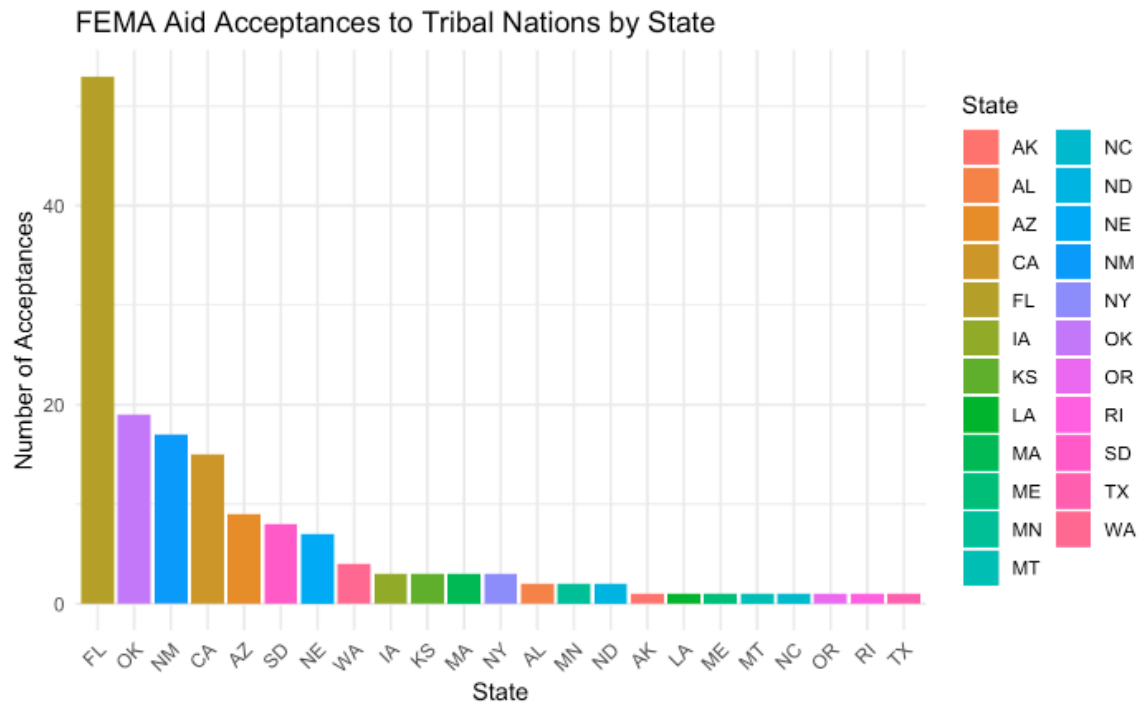


Figure 6.2. Examining the number of acceptances and denials to tribal nations over time, segmented by state, allows for a broader overview of where aid is being requested. Florida leads aid acceptances to tribal nations, and South Dakota leads aid denials.

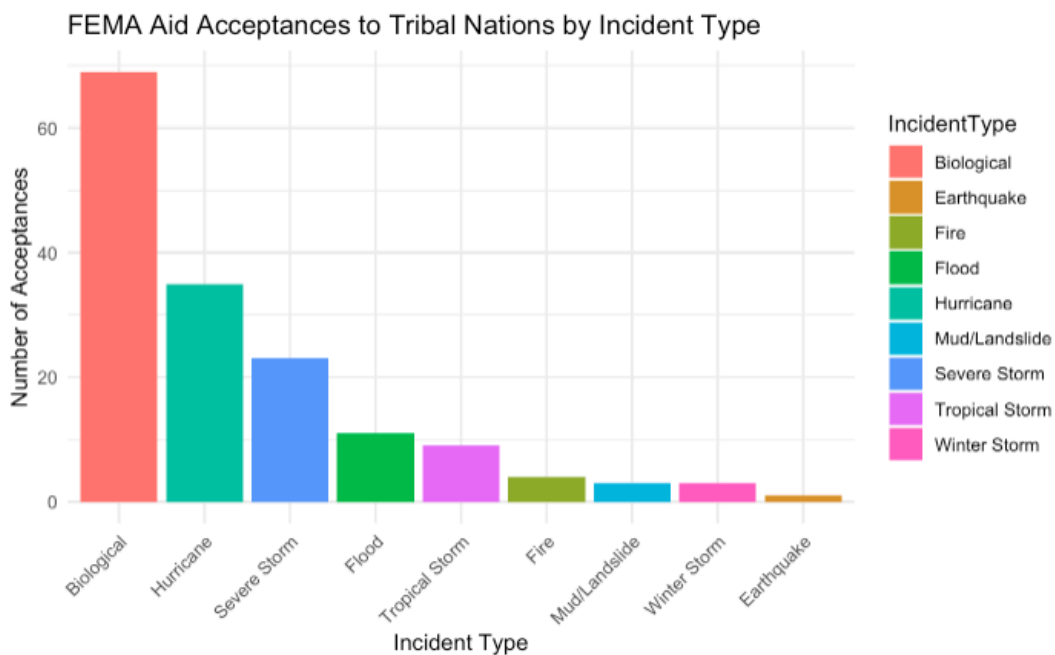


Figure 7.1

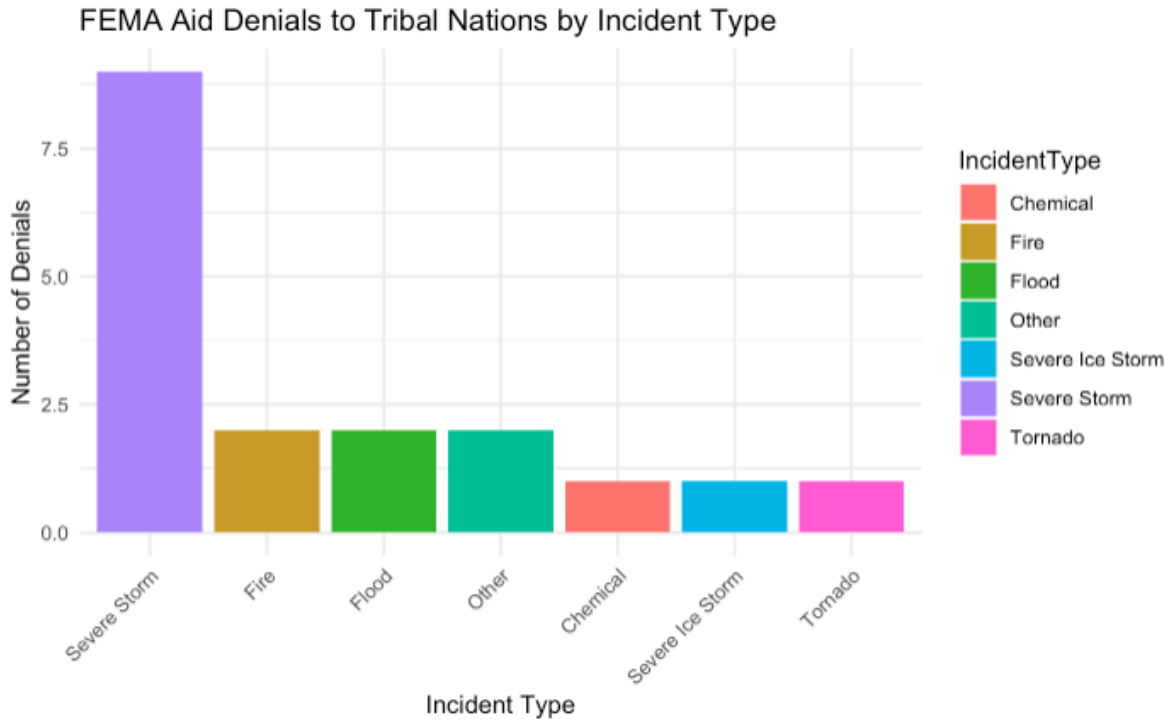


Figure 7.2. Examining the number of acceptances and denials to tribal nations over time, segmented by incident type, allows for a broader overview of why aid is being requested. Biological incidents are most frequently rewarded aid, and Severe Storm incidents are most frequently denied aid.

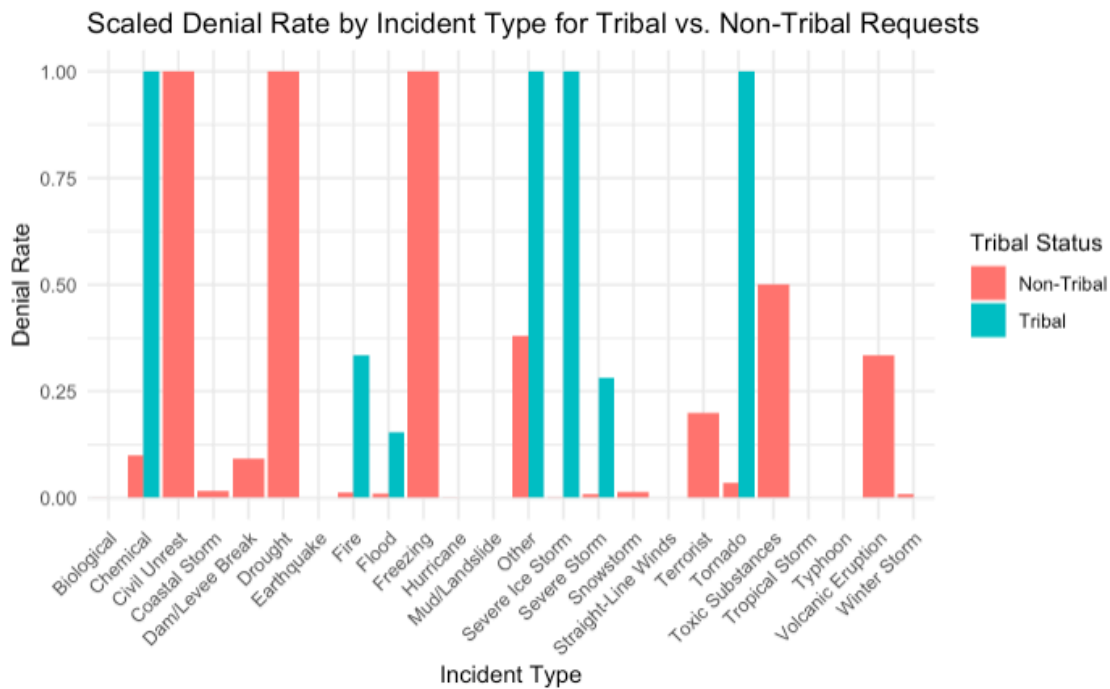


Figure 8.1

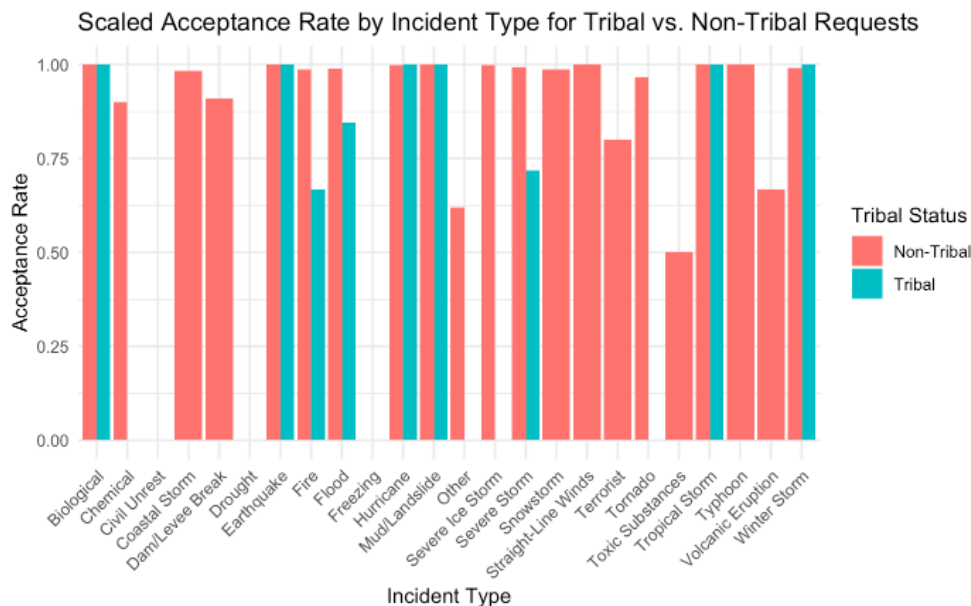


Figure 8.2. Examining the scaled denial rate by incident type for tribal vs. non-tribal requests, tribal requests are denied at a higher rate than non-tribal requests for all incident types for which both types of requests have been made.

```
Call:
glm(formula = requestResult_binary ~ ., family = binomial(link = "logit"),
    data = train_data)
```

Coefficients:

	Estimate	Std. Error	z value	Pr(> z)
(Intercept)	-0.204704	0.564341	-0.363	0.7168
tribalRequest	-2.642508	1.065964	-2.479	0.0132 *
incidentType_numeric	0.024995	0.029465	0.848	0.3963
state_numeric	0.017905	0.009738	1.839	0.0660 .
region	-0.064056	0.070287	-0.911	0.3621

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

Null deviance: 266.17 on 191 degrees of freedom
Residual deviance: 246.06 on 187 degrees of freedom
AIC: 256.06

Number of Fisher Scoring iterations: 5

Figure 9. Initial logistic regression model call, including all four parameters.

```

Step: AIC=253.63
requestResult_binary ~ tribalRequest + state_numeric

              Df Deviance   AIC
<none>                247.63 253.63
- state_numeric  1    251.77 255.77
- tribalRequest  1    262.47 266.47

Call:
glm(formula = requestResult_binary ~ tribalRequest + state_numeric,
     family = binomial(link = "logit"), data = train_data)

Coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept)  -0.433344   0.325481  -1.331  0.18306
tribalRequest -2.848205   1.051010  -2.710  0.00673 **
state_numeric  0.019251   0.009576   2.010  0.04438 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

Null deviance: 266.17 on 191 degrees of freedom
Residual deviance: 247.63 on 189 degrees of freedom
AIC: 253.63

Number of Fisher Scoring iterations: 5

```

Figure 10. Final output from backwards stepwise selection on the initial logistic regression model.

```

predicted_classes
  0  1
0 22 18
1 17 23

```

Figure 11. Confusion matrix for the logistic regression model, after performing backwards stepwise selection.

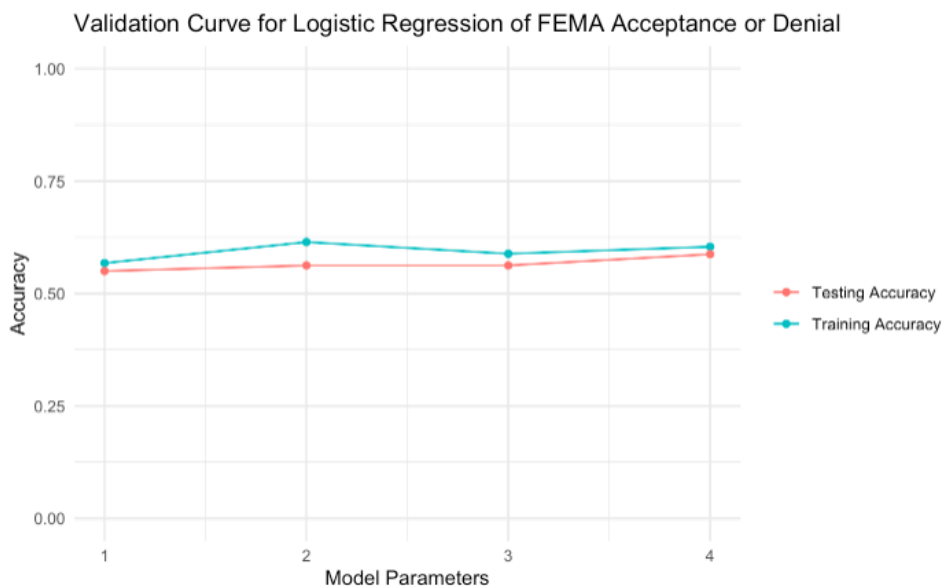


Figure 12

Confusion Matrix and Statistics

```

      Reference
Prediction 0 1
0 21 14
1 19 26

Accuracy : 0.5875
95% CI : (0.4718, 0.6965)
No Information Rate : 0.5
P-Value [Acc > NIR] : 0.07282

Kappa : 0.175

McNemar's Test P-Value : 0.48623

Sensitivity : 0.5250
Specificity : 0.6500
Pos Pred Value : 0.6000
Neg Pred Value : 0.5778
Prevalence : 0.5000
Detection Rate : 0.2625
Detection Prevalence : 0.4375
Balanced Accuracy : 0.5875

'Positive' Class : 0

```

Figure 13. Confusion matrix and summary statistics for the final logistic regression model, including all four original parameters.

Sensitivity	Specificity	Pos Pred Value
0.5250000	0.6500000	0.6000000
Neg Pred Value	Precision	Recall
0.5777778	0.6000000	0.5250000
F1	Prevalence	Detection Rate
0.5600000	0.5000000	0.2625000
Detection Prevalence	Balanced Accuracy	
0.4375000	0.5875000	

Figure 14. Summary statistics for the final logistic regression model.

```

Coefficients:
      Estimate Std. Error z value Pr(>|z|)
(Intercept)   -0.204704   0.564341  -0.363   0.7168
tribalRequest  -2.642508   1.065964  -2.479   0.0132 *
state_numeric    0.017905   0.009738   1.839   0.0660 .
incidentType_numeric 0.024995   0.029465   0.848   0.3963
region         -0.064056   0.070287  -0.911   0.3621
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

Null deviance: 266.17  on 191  degrees of freedom
Residual deviance: 246.06  on 187  degrees of freedom
AIC: 256.06

Number of Fisher Scoring iterations: 5

```

Figure 15. Model summary output for the final logistic regression model.


```
Call:
randomForest(formula = requestResult_binary ~ ., data = train_data, importance = TRUE, ntree = 500)
Type of random forest: classification
Number of trees: 500
No. of variables tried at each split: 2

OOB estimate of error rate: 22.4%
Confusion matrix:
  0 1 class.error
0 76 20 0.2083333
1 23 73 0.2395833
```

Figure 16. Initial random forest call, including all four parameters and 500 trees.

```
Reference
Prediction 0 1
0 31 15
1 9 25

Accuracy : 0.7
95% CI : (0.5872, 0.7974)
No Information Rate : 0.5
P-Value [Acc > NIR] : 0.0002258

Kappa : 0.4

McNemar's Test P-Value : 0.3074342

Sensitivity : 0.7750
Specificity : 0.6250
Pos Pred Value : 0.6739
Neg Pred Value : 0.7353
Prevalence : 0.5000
Detection Rate : 0.3875
Detection Prevalence : 0.5750
Balanced Accuracy : 0.7000

'Positive' Class : 0
```

Figure 17. Confusion matrix and summary statistics for the initial random forest model.

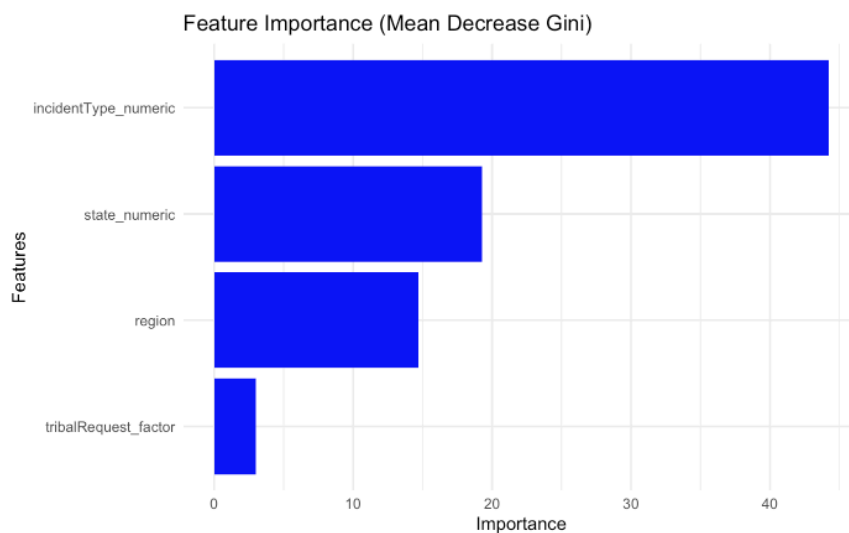


Figure 18. Feature importance for the four features in the initial random forest model.

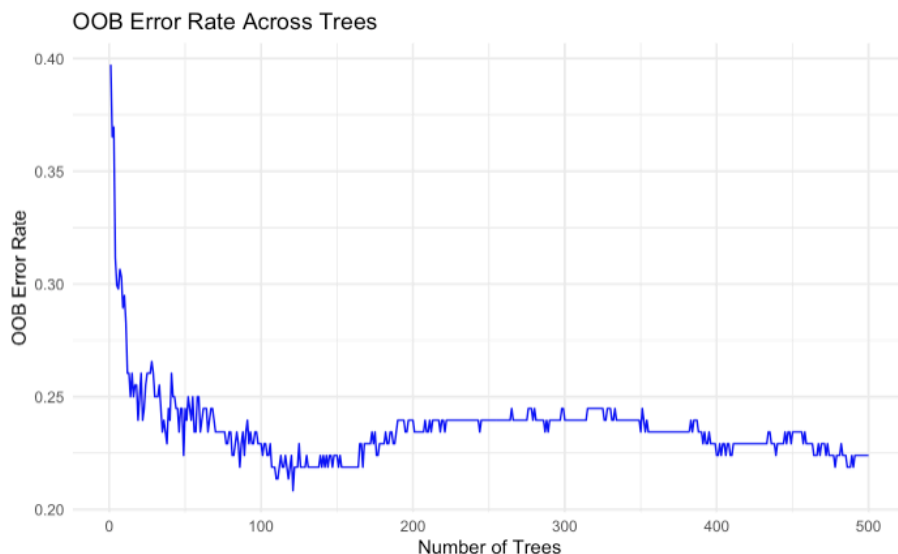


Figure 19. The optimal number of trees to reduce OOB error is around 100 to 125 trees.

```
Call:
randomForest(formula = requestResult_binary ~ ., data = train_data, importance = TRUE, ntree = 125)
Type of random forest: classification
Number of trees: 125
No. of variables tried at each split: 2

OOB estimate of error rate: 23.44%
Confusion matrix:
  0 1 class.error
0 76 20 0.2083333
1 25 71 0.2604167
```

Figure 20. Call for the final random forest model, including all four parameters and 125 trees.

```
Confusion Matrix and Statistics

      Reference
Prediction 0 1
 0 32 15
 1  8 25

Accuracy : 0.7125
95% CI : (0.6005, 0.8082)
No Information Rate : 0.5
P-Value [Acc > NIR] : 9.156e-05

Kappa : 0.425

McNemar's Test P-Value : 0.2109

Sensitivity : 0.8000
Specificity : 0.6250
Pos Pred Value : 0.6809
Neg Pred Value : 0.7576
Prevalence : 0.5000
Detection Rate : 0.4000
Detection Prevalence : 0.5875
Balanced Accuracy : 0.7125

'Positive' Class : 0
```

Figure 21. Confusion matrix and summary statistics for the final random forest model.

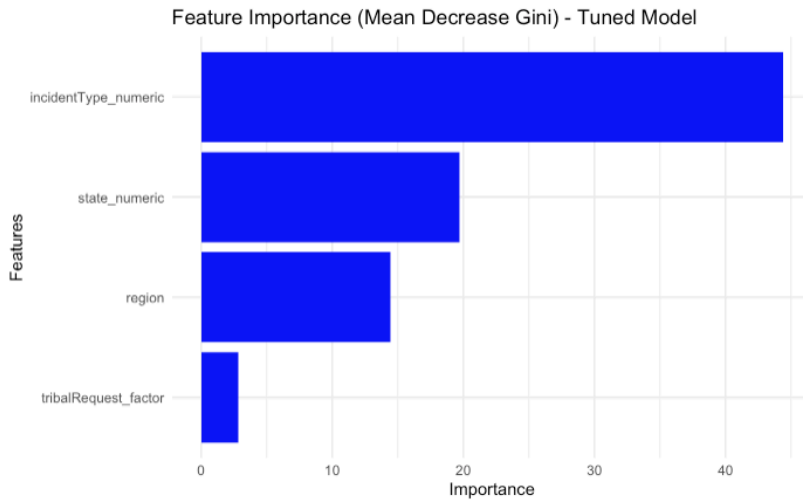


Figure 22. Feature importances for the four features in the final random forest model.

DATA200 Final Project

Preliminary Work

```
# Load necessary libraries
library(tidyverse)
```

```
## — Attaching core tidyverse packages — tidyverse 2.0.0 —
## ✓ dplyr      1.1.4      ✓ readr      2.1.5
## ✓ forcats    1.0.0      ✓ stringr    1.5.1
## ✓ ggplot2    3.5.1      ✓ tibble     3.2.1
## ✓ lubridate  1.9.3      ✓ tidyr      1.3.1
## ✓ purrr      1.0.2
## — Conflicts — tidyverse_conflicts() —
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()     masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library(ggplot2)
library(dplyr)
library(caret)
```

```
## Loading required package: lattice
##
## Attaching package: 'caret'
##
## The following object is masked from 'package:purrr':
##
##     lift
```

```
# Set working directory -> change path url as needed!!!
# *if this isn't working, manually import data.
setwd('/Users/hannahmarr/Desktop/Tufts/DATA200/Final_Project_200/')

# Import data
acceptances <- read.csv("DisasterDeclarationsSummaries.csv")
denials <- read.csv("DeclarationDenials.csv")
```

Exploratory Analysis

```
#### FOR 'acceptances' DATASET ####
# Display the first few rows of the dataframe to inspect the data
head(acceptances)
```

femaDeclarationString <chr>	disasterNumber <int>	state <chr>	declarationType <chr>	declarationDate <chr>
1 FM-5530-NV	5530	NV	FM	2024-08-12T00:00:00.000Z
2 FM-5529-OR	5529	OR	FM	2024-08-09T00:00:00.000Z
3 FM-5528-OR	5528	OR	FM	2024-08-06T00:00:00.000Z
4 FM-5527-OR	5527	OR	FM	2024-08-02T00:00:00.000Z
5 FM-5526-CO	5526	CO	FM	2024-08-01T00:00:00.000Z
6 FM-5525-CO	5525	CO	FM	2024-07-31T00:00:00.000Z

6 rows | 1-7 of 29 columns

```
# Get the dimensions of the dataframe (number of rows and columns).
dim(acceptances)
```

```
## [1] 67245    28
```

```
# Check for missing values in each column.
colSums(is.na(acceptances))
```

```
##      femaDeclarationString      disasterNumber      state
##              0              0              0
##      declarationType      declarationDate      fyDeclared
##              0              0              0
##      incidentType      declarationTitle      ihProgramDeclared
##              0              0              0
##      iaProgramDeclared      paProgramDeclared      hmProgramDeclared
##              0              0              0
##      incidentBeginDate      incidentEndDate      disasterCloseoutDate
##              0              0              0
##      tribalRequest      fipsStateCode      fipsCountyCode
##              0              0              0
##      placeCode      designatedArea      declarationRequestNumber
##              0              0              0
##      lastIAFilingDate      incidentId      region
##              0              0              0
##      designatedIncidentTypes      lastRefresh      hash
##              0              0              0
##      id
##              0
```

```
#### FOR 'denials' DATASET #####
# Display the first few rows of the dataframe to inspect the data
head(denials)
```

	declarationRequestNumber <int>	region <int>	stateAbbreviation <chr>	state <chr>	tribalRequest <int>
1	23	10	ID	Idaho	0
2	40	6	OK	Oklahoma	0
3	46	6	LA	Louisiana	0
4	51	4	NC	North Carolina	0
5	56	5	IL	Illinois	0
6	79	6	TX	Texas	0

6 rows | 1-6 of 21 columns

```
# Get the dimensions of the dataframe (number of rows and columns).
dim(denials)
```

```
## [1] 317 20
```

```
# Check for missing values in each column.
colSums(is.na(denials))
```

```
## declarationRequestNumber      region
##                0                0
##      stateAbbreviation      state
##                0                0
##      tribalRequest      declarationRequestDate
##                0                0
##      declarationRequestType      incidentName
##                0                0
##      requestedIncidentTypes      requestedIncidentBeginDate
##                0                0
##      requestedIncidentEndDate      currentRequestStatus
##                0                0
##      requestStatusDate      ihProgramRequested
##                0                0
##      iaProgramRequested      paProgramRequested
##                0                0
##      hmProgramRequested      incidentId
##                0                0
##      incidentBeginDate      id
##                0                0
```

We will not immediately drop rows with null values, since values in other columns in these rows may become applicable, while the columns with null values do not need to be included in analysis.

Cleaning the Data

First, we did a manual inspection of the column names of the 'denials' dataset and adjusted them so that the two datasets could be merged

```
denials <- select(denials, -state) #orig, 'state' was full name not abbreviation
```

```
denials <- rename(denials,
                  state = stateAbbreviation,
                  declarationDate = declarationRequestDate,
                  declarationType = declarationRequestType,
                  declarationTitle = incidentName,
                  incidentType = requestedIncidentTypes,
                  incidentEndDate = requestedIncidentEndDate,
                  )
```

```
str(denials)
```

```
## 'data.frame':   317 obs. of  19 variables:
## $ declarationRequestNumber : int  23 40 46 51 56 79 93 138 154 155 ...
## $ region                  : int  10 6 6 4 5 6 1 6 6 5 ...
## $ state                   : chr  "ID" "OK" "LA" "NC" ...
## $ tribalRequest           : int  0 0 0 0 0 0 0 0 0 0 ...
## $ declarationDate         : chr  "2000-03-03T00:00:00.000Z" "2000-05-10T00:00:00.0
00Z" "2000-06-05T00:00:00.000Z" "2000-06-06T00:00:00.000Z" ...
## $ declarationType         : chr  "Major Disaster" "Major Disaster" "Major Disaste
r" "Major Disaster" ...
## $ declarationTitle        : chr  "ID-ROCKSLIDES-02-11-2000" "OK-Tulsa Flooding-5-6
-00" "LA-Tornado-North Louisiana-4/23/00" "NC -Severe Storms and Possible Tornadoes" ...
## $ incidentType            : chr  "Mud/Landslide" "Flood" "Tornado" "Severe Storm"
...
## $ requestedIncidentBeginDate: chr  "2000-01-30T00:00:00.000Z" "2000-05-05T00:00:00.0
00Z" "2000-04-23T00:00:00.000Z" "2000-05-25T00:00:00.000Z" ...
## $ incidentEndDate         : chr  "" "2000-05-08T00:00:00.000Z" "2000-04-24T00:00:0
0.000Z" "2000-05-25T00:00:00.000Z" ...
## $ currentRequestStatus    : chr  "Turndown" "Turndown" "Turndown" "Turndown" ...
## $ requestStatusDate       : chr  "2000-03-17T00:00:00.000Z" "2000-05-24T00:00:00.0
00Z" "2000-06-21T00:00:00.000Z" "2000-06-23T00:00:00.000Z" ...
## $ ihProgramRequested      : int  0 0 0 0 0 0 0 0 0 0 ...
## $ iaProgramRequested      : int  1 1 1 0 0 0 1 0 0 0 ...
## $ paProgramRequested      : int  0 0 0 1 1 1 0 1 1 1 ...
## $ hmProgramRequested      : int  1 1 1 1 1 0 1 0 0 1 ...
## $ incidentId              : int  2000021103 2000050701 2000042401 2000052601 20000
51901 2000073101 2000081404 2000090803 2000092205 2000091401 ...
## $ incidentBeginDate       : chr  "2000-01-30T00:00:00.000Z" "2000-05-06T00:00:00.0
00Z" "2000-04-23T00:00:00.000Z" "2000-05-25T00:00:00.000Z" ...
## $ id                      : chr  "61ebab68-adda-44c9-bca0-7c3b7a87c40f" "cf5fd28e-
0195-43be-9bbf-9f153dcc5b40" "779546f1-d286-45de-95f3-ff8905723a9f" "9c6e0548-dd06-48f5-
9d7a-32dc6ccc24d5" ...
```

```
str(acceptances)
```

```
## 'data.frame':    67245 obs. of  28 variables:
## $ femaDeclarationString : chr "FM-5530-NV" "FM-5529-OR" "FM-5528-OR" "FM-5527-OR"
...
## $ disasterNumber       : int  5530 5529 5528 5527 5526 5525 5525 5524 5523 5522
...
## $ state                : chr  "NV" "OR" "OR" "OR" ...
## $ declarationType      : chr  "FM" "FM" "FM" "FM" ...
## $ declarationDate      : chr  "2024-08-12T00:00:00.000Z" "2024-08-09T00:00:00.000Z"
"2024-08-06T00:00:00.000Z" "2024-08-02T00:00:00.000Z" ...
## $ fyDeclared          : int   2024 2024 2024 2024 2024 2024 2024 2024 2024 2024
...
## $ incidentType        : chr  "Fire" "Fire" "Fire" "Fire" ...
## $ declarationTitle     : chr  "GOLD RANCH FIRE" "LEE FALLS FIRE" "ELK LANE FIRE"
"MILE MARKER 132 FIRE" ...
## $ ihProgramDeclared   : int    0 0 0 0 0 0 0 0 0 0 ...
## $ iaProgramDeclared   : int    0 0 0 0 0 0 0 0 0 0 ...
## $ paProgramDeclared   : int    1 1 1 1 1 1 1 1 1 1 ...
## $ hmProgramDeclared   : int    1 1 1 1 1 1 1 1 1 1 ...
## $ incidentBeginDate   : chr  "2024-08-11T00:00:00.000Z" "2024-08-08T00:00:00.000Z"
"2024-08-04T00:00:00.000Z" "2024-08-02T00:00:00.000Z" ...
## $ incidentEndDate     : chr  "" "" "" "" ...
## $ disasterCloseoutDate : chr  "" "" "" "" ...
## $ tribalRequest       : int    0 0 0 0 0 0 0 0 0 0 ...
## $ fipsStateCode       : int   32 41 41 41 8 8 8 8 56 6 ...
## $ fipsCountyCode      : int   31 67 31 17 59 13 69 69 31 29 ...
## $ placeCode           : int  99031 99067 99031 99017 99059 99013 99069 99069 990
31 99029 ...
## $ designatedArea      : chr  "Washoe (County)" "Washington (County)" "Jefferson
(County)" "Deschutes (County)" ...
## $ declarationRequestNumber: int  24123 24122 24116 24111 24106 24105 24105 24104 241
03 24102 ...
## $ lastIAFilingDate    : chr  "" "" "" "" ...
## $ incidentId          : num  2.02e+09 2.02e+09 2.02e+09 2.02e+09 2.02e+09 ...
## $ region              : int   9 10 10 10 8 8 8 8 8 9 ...
## $ designatedIncidentTypes : chr  "R" "R" "R" "R" ...
## $ lastRefresh         : chr  "2024-08-27T18:22:14.800Z" "2024-08-27T18:22:14.800Z"
"2024-08-27T18:22:14.800Z" "2024-08-27T18:22:14.800Z" ...
## $ hash                : chr  "5d07e7c51bb300bfbec94a699a1e1ab1d61a97cd" "ae87cf3
c6ed795015b714af7166c7c295b2b67c7" "432cf0995c47e3895cea696ede5621b810460501" "2f21d90cb
6bc64b0d4121aa3f18d852bbb4b11fa" ...
## $ id                  : chr  "f15a7a79-f1c3-41bb-8a5c-c05fbae34423" "09e3f81a-5e
16-4b72-b317-1c64e0cfa59c" "59983f89-30bf-4888-b21b-62e8d57d9aac" "8d13ecf0-bc2f-496b-8c
9f-b2e73da832a0" ...
```

Merging the Datasets


```
# Merging the two datasets
```

```
data <- merge(denials, acceptances, by = c("state",
                                           "tribalRequest",
                                           "declarationDate",
                                           "declarationTitle",
                                           "incidentType",
                                           "incidentEndDate",
                                           "incidentId",
                                           "incidentBeginDate",
                                           "id",
                                           "region",
                                           "declarationType"), all = TRUE)
```

```
# Inspect the dataset to determine if the merge was successful
```

```
head(data)
```

state <chr>	tribalRequest <int>	declarationDate <chr>	declarationTitle <chr>	incidentType <chr>
1 AK	0	1953-10-30T00:00:00.000Z	SEVERE HARDSHIP	Other
2 AK	0	1954-11-10T00:00:00.000Z	SEVERE HARDSHIP	Other
3 AK	0	1955-12-22T00:00:00.000Z	SEVERE HARDSHIP	Other
4 AK	0	1964-03-28T00:00:00.000Z	EARTHQUAKE	Earthquake
5 AK	0	1967-08-17T00:00:00.000Z	SEVERE STORMS & FLOODING	Flood
6 AK	0	1969-12-19T00:00:00.000Z	HEAVY RAINS & LANDSLIDE	Severe Storm

6 rows | 1-6 of 37 columns

```
dim(data)
```

```
## [1] 67562    36
```

```
str(data)
```

```
## 'data.frame':    67562 obs. of  36 variables:
## $ state                : chr  "AK" "AK" "AK" "AK" ...
## $ tribalRequest        : int   0 0 0 0 0 0 0 0 0 0 ...
## $ declarationDate      : chr  "1953-10-30T00:00:00.000Z" "1954-11-10T00:00:00.000Z" "1955-12-22T00:00:00.000Z" "1964-03-28T00:00:00.000Z" ...
## $ declarationTitle     : chr  "SEVERE HARDSHIP" "SEVERE HARDSHIP" "SEVERE HARDSHIP" "EARTHQUAKE" ...
## $ incidentType         : chr  "Other" "Other" "Other" "Earthquake" ...
## $ incidentEndDate      : chr  "1953-10-30T00:00:00.000Z" "1954-11-10T00:00:00.000Z" "1955-12-22T00:00:00.000Z" "1964-03-28T00:00:00.000Z" ...
## $ incidentId           : num   53012 54019 55017 64015 67024 ...
## $ incidentBeginDate    : chr  "1953-10-30T00:00:00.000Z" "1954-11-10T00:00:00.000Z" "1955-12-22T00:00:00.000Z" "1964-03-28T00:00:00.000Z" ...
## $ id                   : chr  "353cdf38-9ad2-40cd-b92a-58976c13236f" "6cf23fc4-1cd6-4d3d-8dd5-154ab481d09f" "8a2aae98-dadd-4b5d-b449-ae238dba5d3" "6fba469d-7108-4c1e-b82d-60548ede4178" ...
## $ region               : int   10 10 10 10 10 10 10 10 10 10 ...
## $ declarationType      : chr  "DR" "DR" "DR" "DR" ...
## $ declarationRequestNumber.x: int   NA NA NA NA NA NA NA NA NA NA ...
## $ requestedIncidentBeginDate: chr  NA NA NA NA ...
## $ currentRequestStatus  : chr  NA NA NA NA ...
## $ requestStatusDate     : chr  NA NA NA NA ...
## $ ihProgramRequested    : int   NA NA NA NA NA NA NA NA NA NA ...
## $ iaProgramRequested    : int   NA NA NA NA NA NA NA NA NA NA ...
## $ paProgramRequested    : int   NA NA NA NA NA NA NA NA NA NA ...
## $ hmProgramRequested    : int   NA NA NA NA NA NA NA NA NA NA ...
## $ femaDeclarationString : chr  "DR-13-AK" "DR-31-AK" "DR-46-AK" "DR-168-AK" ...
## $ disasterNumber        : int   13 31 46 168 230 281 2001 2004 2005 2006 ...
## $ fyDeclared            : int   1954 1955 1956 1964 1967 1970 1970 1971 1971 1973 ...
## $ ihProgramDeclared     : int   0 0 0 0 0 0 0 0 0 0 ...
## $ iaProgramDeclared     : int   1 1 1 1 1 1 0 0 0 0 ...
## $ paProgramDeclared     : int   1 1 1 1 1 1 1 1 1 1 ...
## $ hmProgramDeclared     : int   1 1 1 1 0 0 0 0 0 0 ...
## $ disasterCloseoutDate  : chr  "1957-09-01T00:00:00.000Z" "1957-09-01T00:00:00.000Z" "1956-12-01T00:00:00.000Z" "1971-06-17T00:00:00.000Z" ...
## $ fipsStateCode         : int   2 2 2 2 2 2 2 2 2 2 ...
## $ fipsCountyCode        : int   0 0 0 0 0 0 0 0 0 0 ...
## $ placeCode             : int   0 0 0 0 75165 75165 0 0 0 0 ...
## $ designatedArea        : chr  "Statewide" "Statewide" "Statewide" "Statewide" ...
## $ declarationRequestNumber.y: int   53012 54019 55017 64015 67024 69044 70100 71042 71044 73050 ...
## $ lastIAFilingDate      : chr  "" "" "" "" ...
## $ designatedIncidentTypes : chr  "" "" "" "" ...
## $ lastRefresh           : chr  "2024-08-27T18:22:14.800Z" "2024-08-27T18:22:14.800Z" "2024-08-27T18:22:14.800Z" "2024-08-27T18:22:14.800Z" ...
## $ hash                  : chr  "9c2d55e534115516cf862eed47585d695cd1e24" "8aa70d2832920cf1023d7e14d7a03160f6f191ae" "552c686cac2fba0094e5fc89069778b4a8fd1fd3" "4ab30e317c3672060e315922b972311ec0714ff9" ...
```

More Data Cleaning

We will add a column that designates from which dataset the data came. This column will encode the data as 'Denial' or 'Acceptance', based on if it was in the DeclarationDenials dataset ('denials') or the DisasterDeclarationSummaries dataset ('acceptances').

```
# Create the new column 'requestResult'
data$requestResult <- ifelse(is.na(data$paProgramRequested),
                             'Acceptance',
                             'Denial')
```

Denials data only goes back to 2000, and tribal nations were only allowed to submit their own declaration requests to FEMA in January 2013. Therefore, we will drop all data from 2012 and prior.

```
# Converting the incidentBeginDate column to a date-time format and remove rows with data from before the year 2013.
data$incidentBeginDate <- as.POSIXct(data$incidentBeginDate, format = "%Y-%m-%dT%H:%M:%SZ", tz = "UTC")
data <- data[is.na(data$incidentBeginDate) | format(data$incidentBeginDate, "%Y") >= 2013, ]
```

```
# Inspect data to confirm no issues so far
head(data)
```

state <chr>	tribalRequest <int>	declarationDate <chr>	declarationTitle <chr>	incidentType <chr>
152 AK	0	2013-06-25T00:00:00.000Z	FLOODING	Flood
153 AK	0	2013-06-25T00:00:00.000Z	FLOODING	Flood
154 AK	0	2013-06-25T00:00:00.000Z	FLOODING	Flood
155 AK	0	2013-06-25T00:00:00.000Z	FLOODING	Flood
156 AK	0	2013-06-25T00:00:00.000Z	FLOODING	Flood
157 AK	0	2014-01-16T00:00:00.000Z	FLOODING	Flood

6 rows | 1-6 of 38 columns

```
dim(data)
```

```
## [1] 24439 37
```

Now I will create a new column, declarationMonth, that extracts the month from the declarationDate column to specify the month in which the declaration occurred.

```
data$declarationDate <- as.POSIXct(data$declarationDate, format = "%Y-%m-%dT%H:%M:%SZ",
                                     tz = "UTC")

data$declarationMonth <- format(data$declarationDate, "%m")
```

We will do the same thing for year and create a new column, `declarationYear`, that extracts the year from the `declarationDate` column to specify the year in which the declaration occurred.

```
data$declarationDate <- as.POSIXct(data$declarationDate, format = "%Y-%m-%dT%H:%M:%SZ",
  tz = "UTC")
```

```
data$declarationYear <- format(data$declarationDate, "%Y")
```

```
# View the first few rows and dimensions of the updated dataset
head(data)
```

state <chr>	tribalRequest <int>	declarationDate <dtm>	declarationTitle <chr>	incidentType <chr>	incidentEndDate <chr>
152 AK	0	2013-06-25	FLOODING	Flood	2013-06-11T00:00:
153 AK	0	2013-06-25	FLOODING	Flood	2013-06-11T00:00:
154 AK	0	2013-06-25	FLOODING	Flood	2013-06-11T00:00:
155 AK	0	2013-06-25	FLOODING	Flood	2013-06-11T00:00:
156 AK	0	2013-06-25	FLOODING	Flood	2013-06-11T00:00:
157 AK	0	2014-01-16	FLOODING	Flood	2013-10-28T00:00:

6 rows | 1-7 of 40 columns

```
dim(data)
```

```
## [1] 24439    39
```

```
# Writing dataset to a csv file to save locally
write.csv(data, "data.csv", row.names = FALSE)
```

Visualizing Acceptances v. Denials for FEMA Aid

```
# Create a summary of the 'requestResult' column
result_summary <- table(data$requestResult)

# Convert the table to a data frame for plotting
result_df <- as.data.frame(result_summary)
colnames(result_df) <- c("RequestResult", "Count")

# Create dynamic legend labels with counts
legend_labels <- paste0(result_df$requestResult, " (", result_df$Count, ")")

# Plot the data using ggplot2
ggplot(result_df, aes(x = RequestResult, y = Count, fill = RequestResult)) +
  geom_bar(stat = "identity") +
  labs(
    title = "Comparison of Acceptances vs. Denials of FEMA aid",
    x = "Request Result",
    y = "Count",
    fill = "Request Result"
  ) +
  scale_fill_discrete(labels = legend_labels) + # Add custom legend labels
  theme_minimal()
```

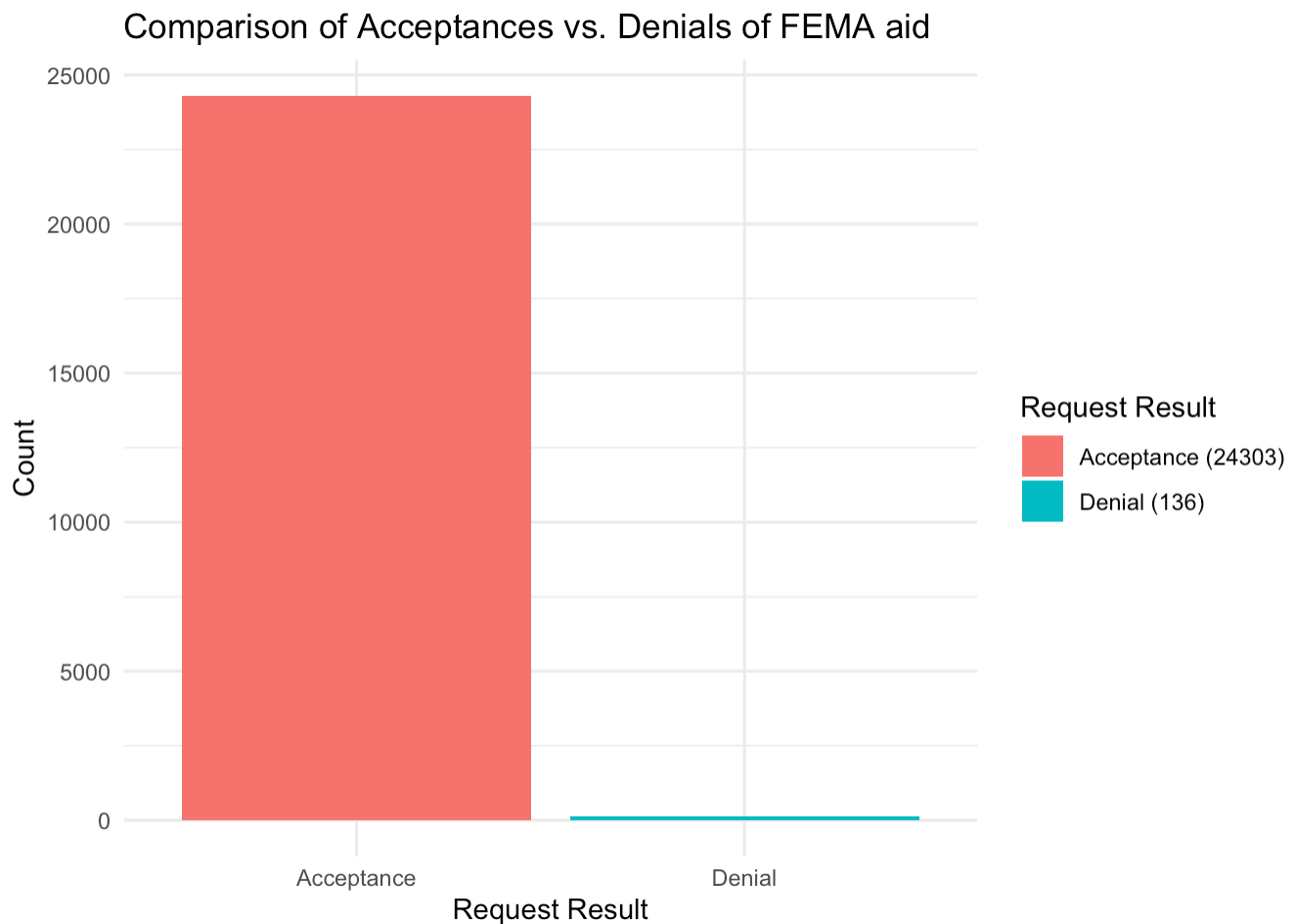


Figure 1. Comparing Overall Acceptances vs. Denials of FEMA aid, we see there are far more documented acceptances than denials of FEMA aid.

Visualizing Variations Among Incident Types

```
# Create a summary of the 'incidentType' column
incident_summary <- table(data$incidentType)

# Convert the summary table to a data frame for visualization
incident_df <- as.data.frame(incident_summary)
colnames(incident_df) <- c("IncidentType", "Count")

# Plot the data using ggplot2
ggplot(incident_df, aes(x = reorder(IncidentType, -Count), y = Count, fill = IncidentType)) +
  geom_bar(stat = "identity") +
  labs(title = "Counts of Each Incident Type", x = "Incident Type", y = "Count") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```

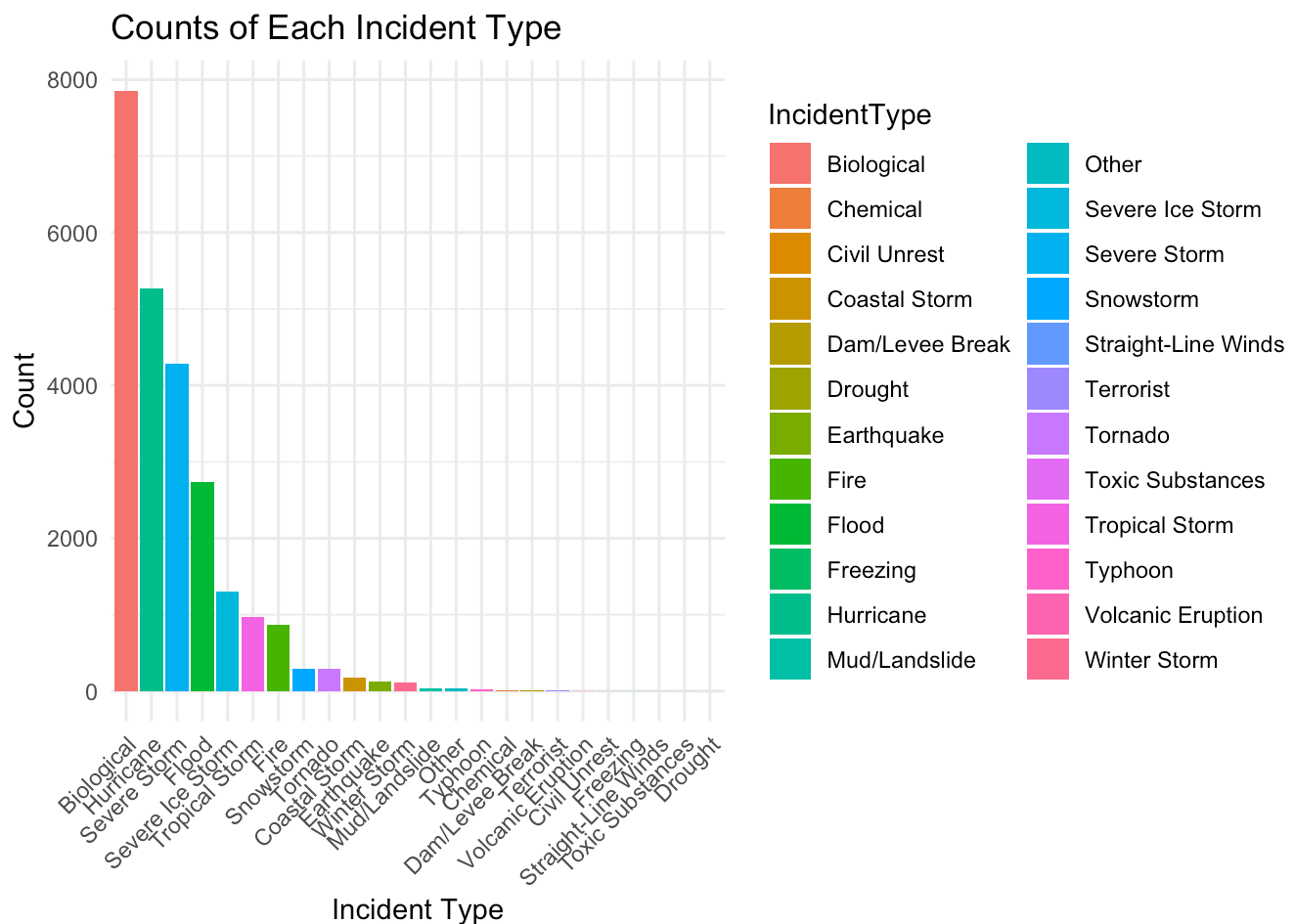


Figure 2. Visualizing the variation among incident types allows us to get a better sense of the various reasons for FEMA aid requests. Biological is the most prevalent incident type, followed by Hurricane, Severe Storm, and Flood.

Examine number of tribal vs. non-tribal nation requests

```
# Create a summary of the tribalRequest column (count 1s and 0s)
tribal_summary <- table(data$tribalRequest)

# Convert the summary table to a data frame for plotting
tribal_df <- as.data.frame(tribal_summary)
colnames(tribal_df) <- c("RequestType", "Count")

# Map 1 to 'Tribal Requests' and 0 to 'Non-Tribal Requests'
tribal_df$RequestType <- factor(tribal_df$RequestType,
                                levels = c(0, 1),
                                labels = c("Non-Tribal Requests", "Tribal Requests"))

# Create custom labels for the legend with counts
custom_labels <- paste(tribal_df$RequestType, " (", tribal_df$Count, ")", sep = "")

# Plot the data using ggplot2
ggplot(tribal_df, aes(x = RequestType, y = Count, fill = RequestType)) +
  geom_bar(stat = "identity") +
  scale_fill_manual(values = c("#4DAF4A", "#377EB8"), labels = custom_labels) +
  labs(title = "Comparison of Tribal vs. Non-Tribal Requests",
       x = "Request Type",
       y = "Count",
       fill = "Request Type (Count)") +
  theme_minimal()
```

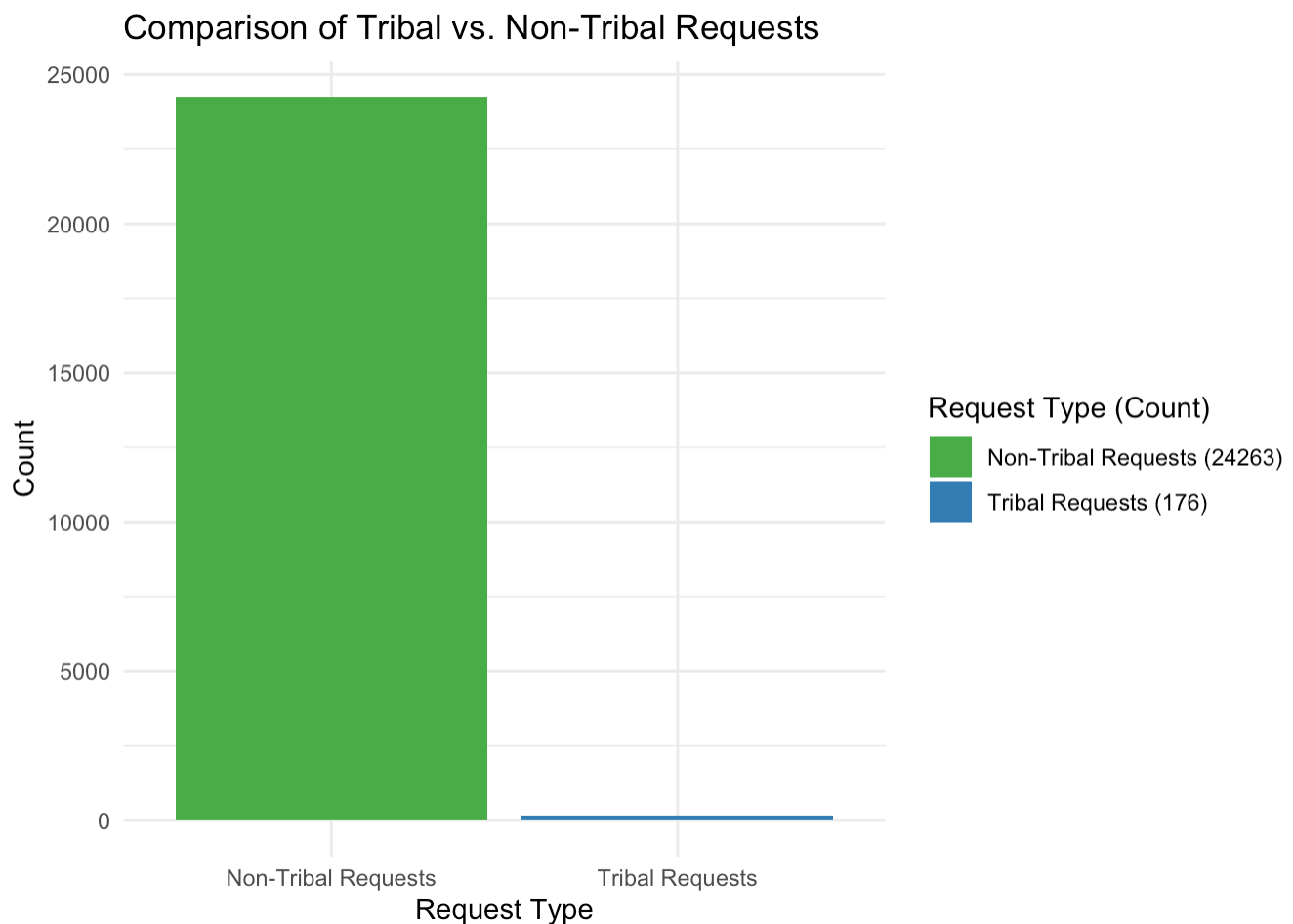


Figure 3. Examining the number of tribal nation vs. non-tribal nation requests shows how infrequent tribal requests are. Part of this divide is likely due to tribal nations not being able to apply for their own declaration request until January 29, 2013. While all pre-2013 data was removed, it is likely that this process is still not as streamlined as the request process for states.

Note that there are very few tribal requests. Later, we will be using LOOCV to ensure results are robust.

Examine number of accepted aid requests for tribal nations over time

```
# Filter data for tribal requests with 'Acceptance' in requestResult
tribal_acceptances <- subset(data, tribalRequest == 1 & requestResult == "Acceptance")

# Count acceptances by year
acceptance_by_year <- table(tribal_acceptances$declarationYear)
acceptance_df <- as.data.frame(acceptance_by_year)
colnames(acceptance_df) <- c("Year", "Count")

# Convert Year to a numeric type for plotting
acceptance_df$Year <- as.numeric(as.character(acceptance_df$Year))

# Plot the data
ggplot(acceptance_df, aes(x = Year, y = Count)) +
  geom_line(color = "blue") +
  geom_point(color = "blue") +
  labs(title = "FEMA Aid Acceptances to Tribal Nations Over Time",
       x = "Year",
       y = "Number of Acceptances") +
  theme_minimal()
```


FEMA Aid Acceptances to Tribal Nations Over Time

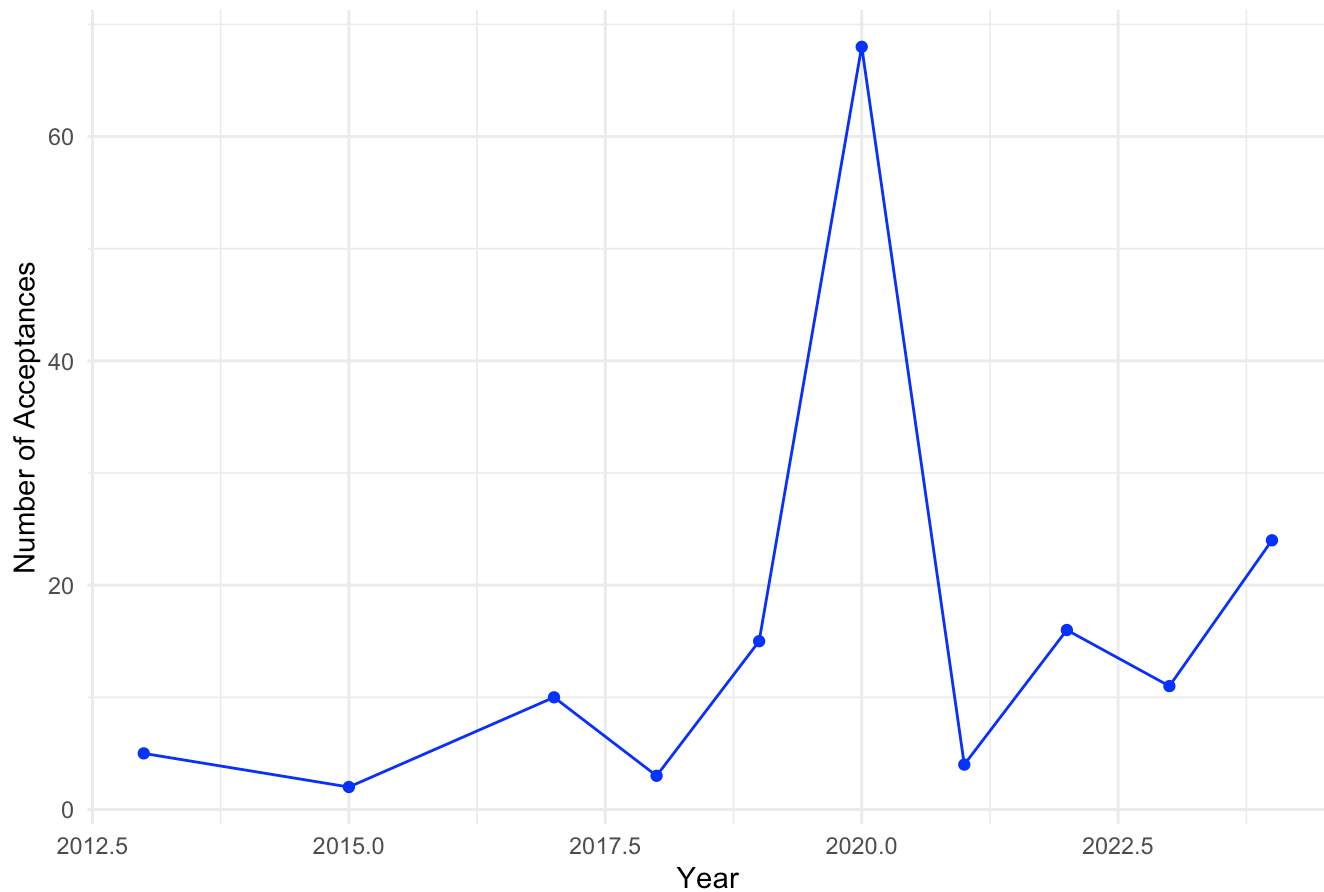


Figure 4.1

Examine number of denied aid requests for tribal nations over time

```
# Filter data for tribal requests with 'Denial' in requestResult
tribal_denials <- subset(data, tribalRequest == 1 & requestResult == "Denial")

# Count denials by year
denials_by_year <- table(tribal_denials$declarationYear)
denials_df <- as.data.frame(denials_by_year)
colnames(denials_df) <- c("Year", "Count")

# Convert Year to a numeric type for plotting
denials_df$Year <- as.numeric(as.character(denials_df$Year))

# Plot the data
ggplot(denials_df, aes(x = Year, y = Count)) +
  geom_line(color = "red") +
  geom_point(color = "red") +
  labs(title = "FEMA Aid Denials to Tribal Nations Over Time",
       x = "Year",
       y = "Number of Denials") +
  theme_minimal()
```

FEMA Aid Denials to Tribal Nations Over Time

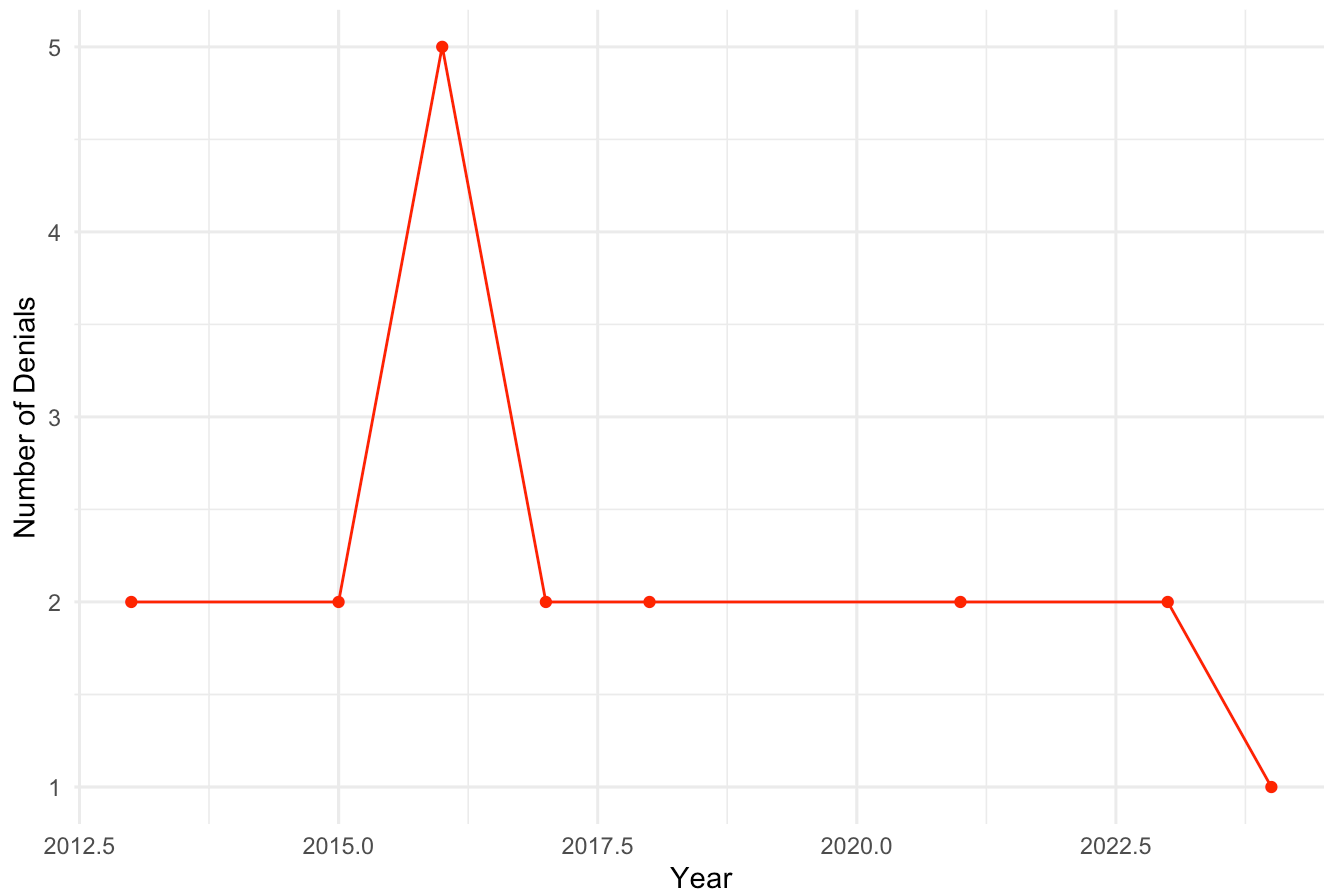


Figure 4.2. Visualizing the number of FEMA aid acceptances over time to tribal nations shows a general upward trajectory, with a sharp spike in 2020, likely due to the COVID-19 pandemic. There appear to be far fewer denials than acceptances.

Examine FEMA aid acceptances to tribal nations over time, segmented by program type declared

```
# Load additional necessary libraries
library(reshape2)
```

```
##
## Attaching package: 'reshape2'
```

```
## The following object is masked from 'package:tidyr':
##
## smiths
```

```

# Filter data for tribal requests only
tribal_data <- subset(data, tribalRequest == 1)

# Aggregate the data by year and program acceptance
program_acceptances <- aggregate(cbind(ihProgramDeclared, iaProgramDeclared, paProgramDe
clared, hmProgramDeclared) ~ declarationYear,
                                data = tribal_data,
                                FUN = sum)

# Melt the data for plotting (long format)
melted_data <- melt(program_acceptances, id.vars = "declarationYear",
                    variable.name = "ProgramType",
                    value.name = "Acceptances")

# Map the column names to more descriptive labels
melted_data$ProgramType <- factor(melted_data$ProgramType,
                                 levels = c("ihProgramDeclared", "iaProgramDeclared",
"paProgramDeclared", "hmProgramDeclared"),
                                 labels = c("Individuals and Households Program Declare
d",
                                           "Individual Assistance Program Declared",
                                           "Public Assistance Program Declared",
                                           "Hazard Mitigation Program Declared"))

# Plot the data
ggplot(melted_data, aes(x = declarationYear, y = Acceptances, color = ProgramType)) +
  geom_line() +
  geom_point() +
  labs(title = "FEMA Aid Acceptances to Tribal Nations Over Time by Program Type",
       x = "Year",
       y = "Number of Acceptances",
       color = "Program Type") +
  theme_minimal()

```

```

## `geom_line()`: Each group consists of only one observation.
## i Do you need to adjust the group aesthetic?

```

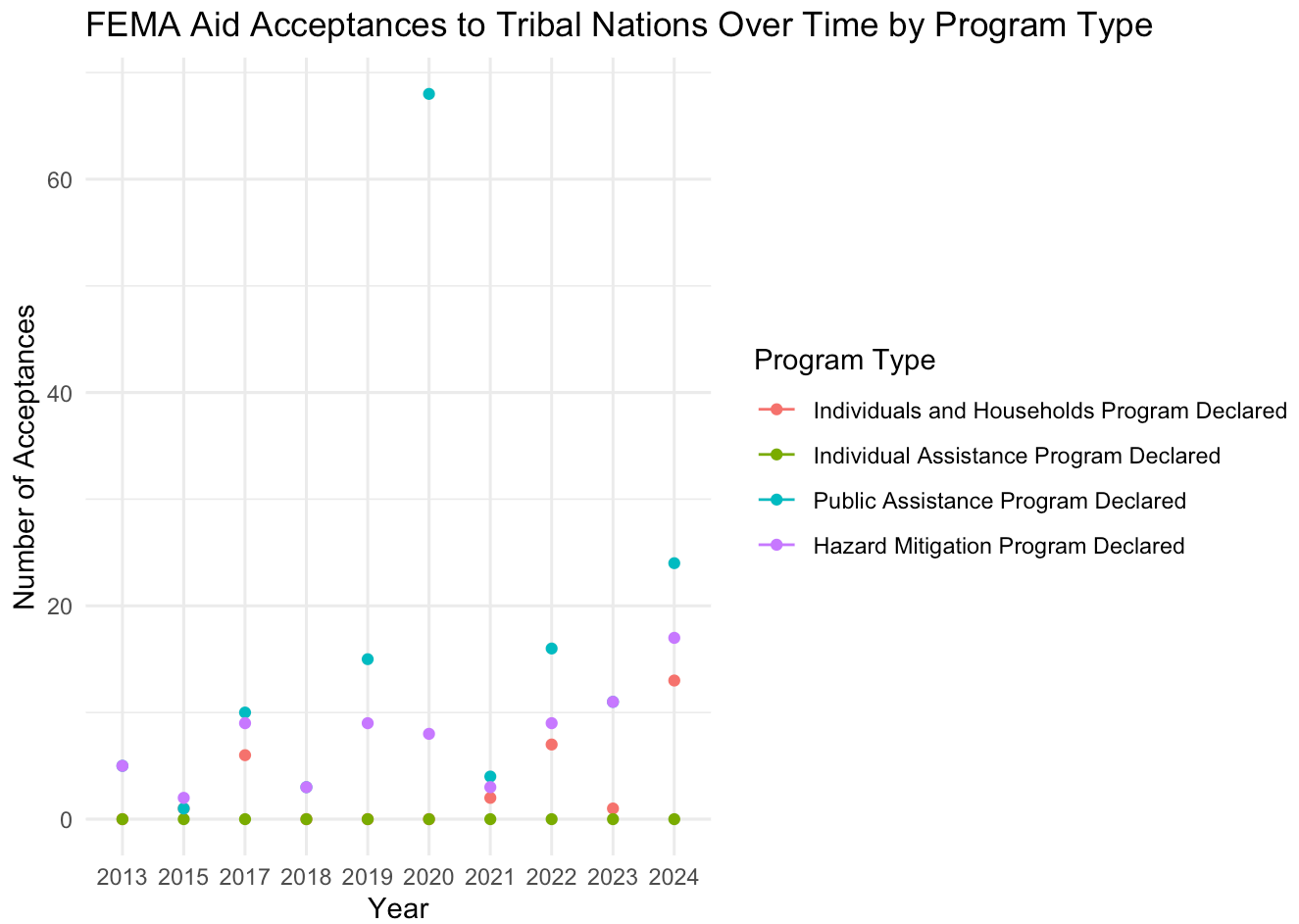


Figure 5.1

Examine FEMA aid denials to tribal nations over time, segmented by program type requested

```

# Filter the data for tribal requests and denials
tribal_denials <- subset(data, tribalRequest == 1 & requestResult == "Denial")

# Aggregate the number of denials by year and program requested
program_denials <- aggregate(cbind(ihProgramRequested, iaProgramRequested, paProgramRequested, hmProgramRequested) ~ declarationYear,
                             data = tribal_denials,
                             FUN = sum)

# Melt the data for plotting (long format)
melted_denials <- melt(program_denials, id.vars = "declarationYear",
                      variable.name = "ProgramType",
                      value.name = "Denials")

# Map the column names to more descriptive labels
melted_denials$ProgramType <- factor(melted_denials$ProgramType,
                                     levels = c("ihProgramRequested", "iaProgramRequested", "paProgramRequested", "hmProgramRequested"),
                                     labels = c("Individuals and Households Program Requested",
                                                "Individual Assistance Program Requested",
                                                "Public Assistance Program Requested",
                                                "Hazard Mitigation Program Requested"))

# Plot the data
ggplot(melted_denials, aes(x = declarationYear, y = Denials, color = ProgramType)) +
  geom_line() +
  geom_point() +
  labs(title = "FEMA Aid Denials to Tribal Nations Over Time by Program Type",
       x = "Year",
       y = "Number of Denials",
       color = "Program Type") +
  theme_minimal()

```

```

## `geom_line()`: Each group consists of only one observation.
## i Do you need to adjust the group aesthetic?

```

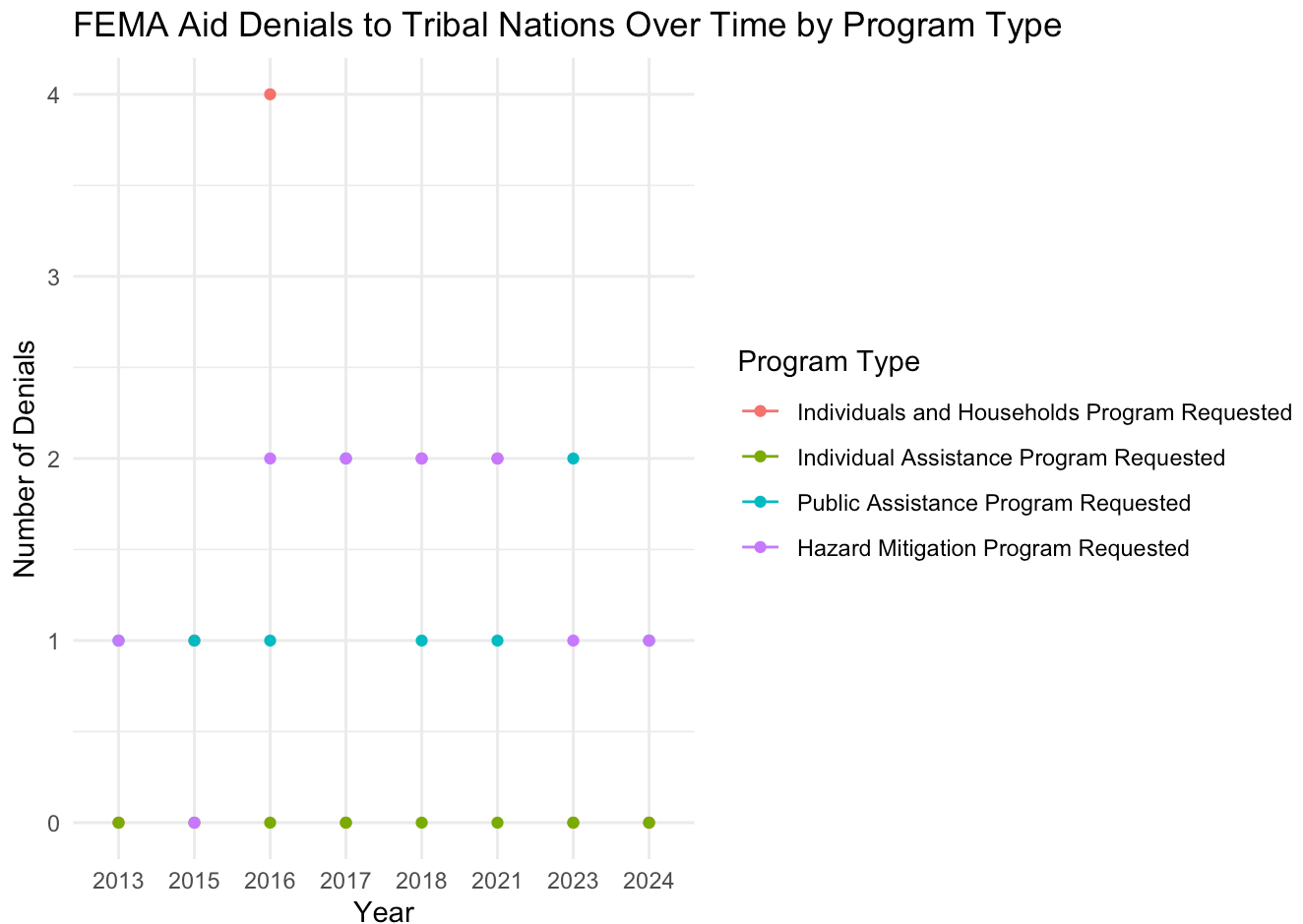


Figure 5.2. Examining the number of acceptances and denials to tribal nations over time, segmented by program type, allows for a broader understanding of what types of aid are being requested.

Examining denials to tribal nations by state

```
# Filter data for tribal requests with 'Denial' in requestResult
tribal_denials <- subset(data, tribalRequest == 1 & requestResult == "Denial")

# Count denials by state
denials_by_state <- table(tribal_denials$state)
denials_df <- as.data.frame(denials_by_state)
colnames(denials_df) <- c("State", "Count")

# Sort the data frame by Count in descending order
denials_df <- denials_df[order(-denials_df$Count), ]

# Plot the data using ggplot2
ggplot(denials_df, aes(x = reorder(State, -Count), y = Count, fill = State)) +
  geom_bar(stat = "identity") +
  labs(title = "FEMA Aid Denials to Tribal Nations by State",
       x = "State",
       y = "Number of Denials") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```

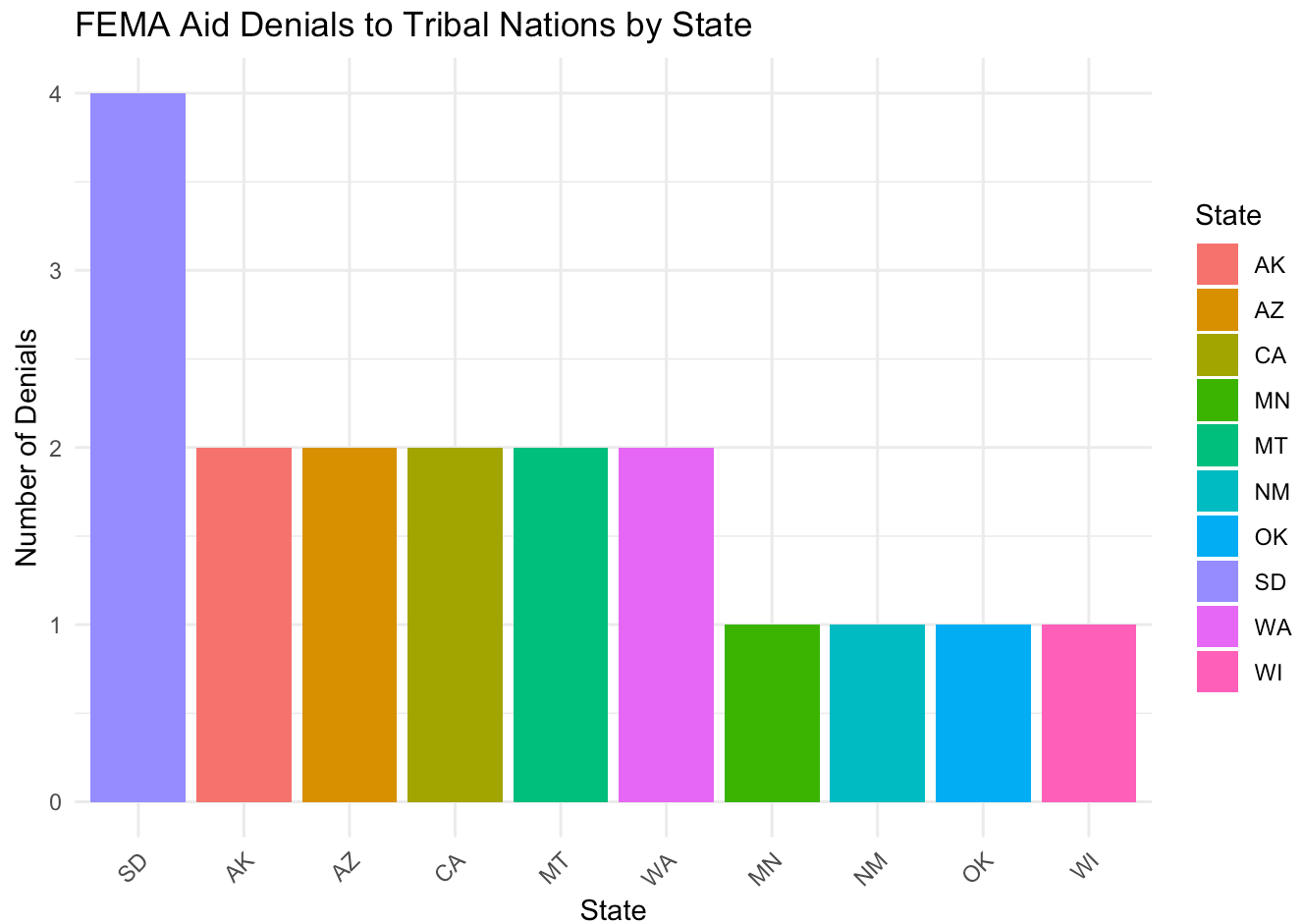


Figure 6.1

Examining acceptances to tribal nation by state

```
# Filter data for tribal requests with 'Acceptance' in requestResult
tribal_acceptances <- subset(data, tribalRequest == 1 & requestResult == "Acceptance")

# Count acceptances by state
acceptances_by_state <- table(tribal_acceptances$state)
acceptances_df <- as.data.frame(acceptances_by_state)
colnames(acceptances_df) <- c("State", "Count")

# Sort the data frame by Count in descending order
acceptances_df <- acceptances_df[order(-acceptances_df$Count), ]

# Plot the data using ggplot2
ggplot(acceptances_df, aes(x = reorder(State, -Count), y = Count, fill = State)) +
  geom_bar(stat = "identity") +
  labs(title = "FEMA Aid Acceptances to Tribal Nations by State",
       x = "State",
       y = "Number of Acceptances") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```

FEMA Aid Acceptances to Tribal Nations by State

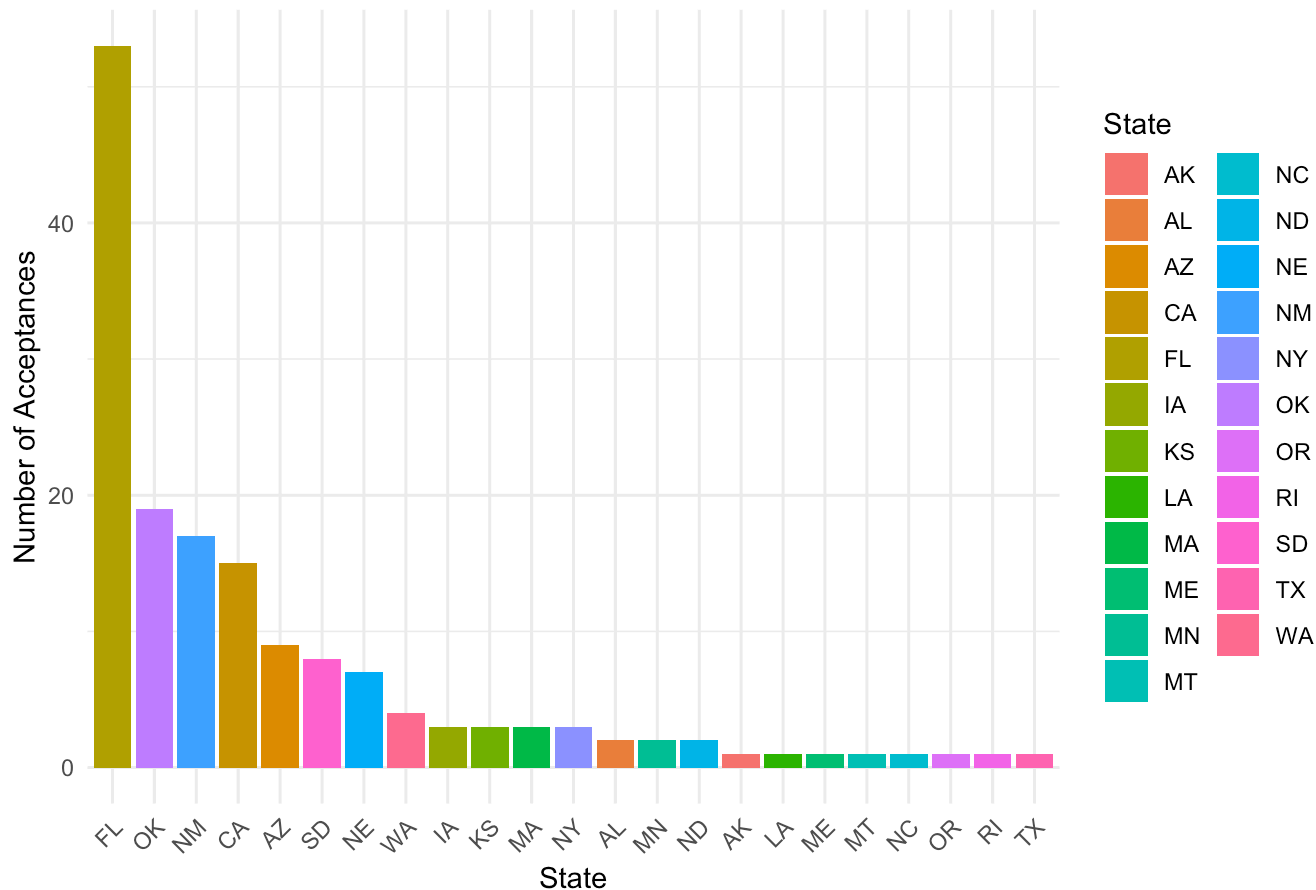


Figure 6.2. Examining the number of acceptances and denials to tribal nations over time, segmented by state, allows for a broader overview of where aid is being requested. Florida leads aid acceptances to tribal nations, and South Dakota leads aid denials.

Examine acceptances to tribal nations by incident type

```
# Filter data for tribal requests with 'Acceptance' in requestResult
tribal_acceptances <- subset(data, tribalRequest == 1 & requestResult == "Acceptance")

# Count acceptances by incident type
acceptances_by_incident <- table(tribal_acceptances$incidentType)
acceptances_df <- as.data.frame(acceptances_by_incident)
colnames(acceptances_df) <- c("IncidentType", "Count")

# Sort the data frame by Count in descending order
acceptances_df <- acceptances_df[order(-acceptances_df$Count), ]

# Plot the data using ggplot2
ggplot(acceptances_df, aes(x = reorder(IncidentType, -Count), y = Count, fill = Incident
Type)) +
  geom_bar(stat = "identity") +
  labs(title = "FEMA Aid Acceptances to Tribal Nations by Incident Type",
       x = "Incident Type",
       y = "Number of Acceptances") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```

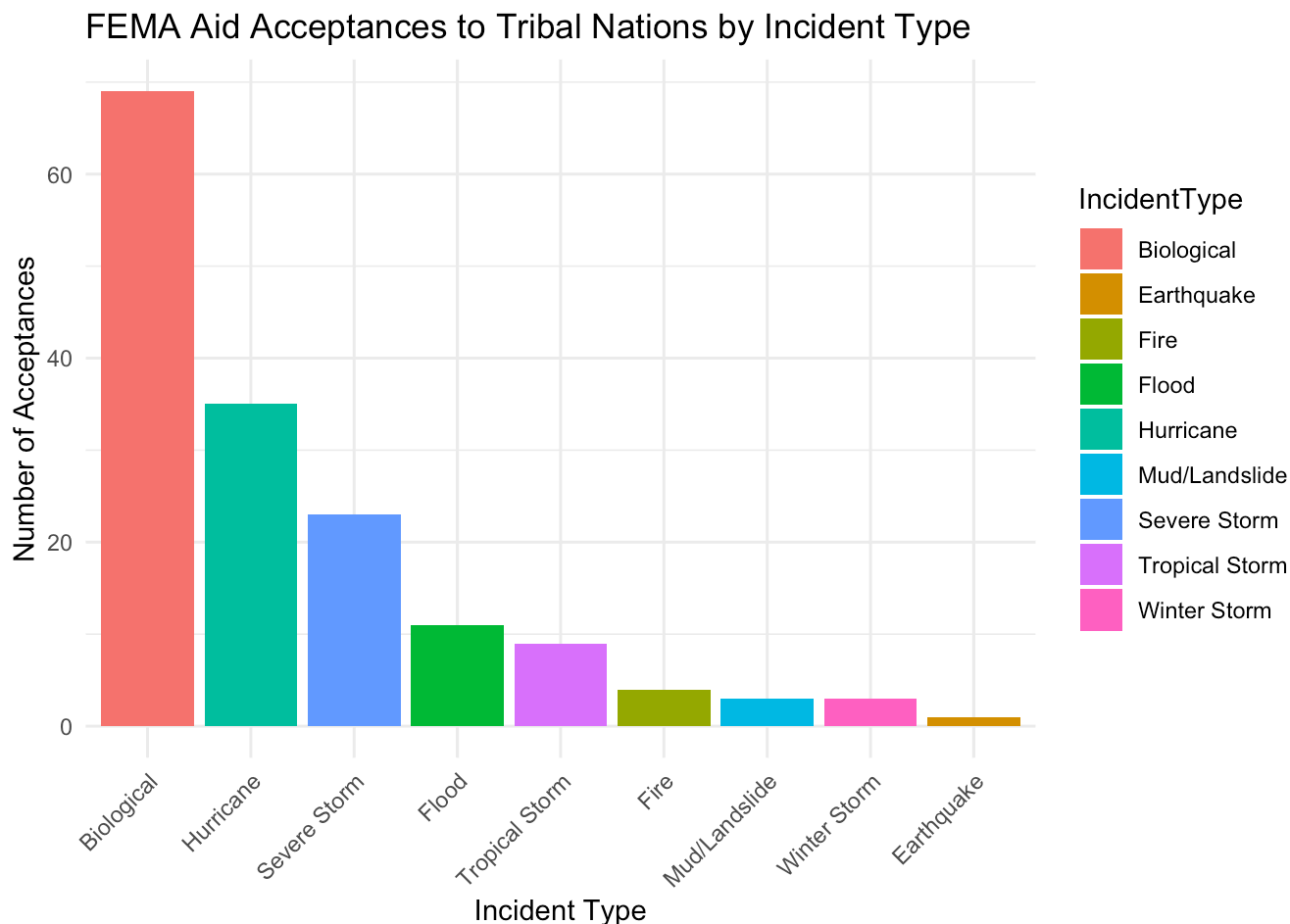



Figure 7.1

Examining denials to tribal nations by incident type

```
# Filter data for tribal requests with 'Denial' in requestResult
tribal_denials <- subset(data, tribalRequest == 1 & requestResult == "Denial")

# Count denials by incident type
denials_by_incident <- table(tribal_denials$incidentType)
denials_df <- as.data.frame(denials_by_incident)
colnames(denials_df) <- c("IncidentType", "Count")

# Sort the data frame by Count in descending order
denials_df <- denials_df[order(-denials_df$Count), ]

# Plot the data using ggplot2
ggplot(denials_df, aes(x = reorder(IncidentType, -Count), y = Count, fill = IncidentType)) +
  geom_bar(stat = "identity") +
  labs(title = "FEMA Aid Denials to Tribal Nations by Incident Type",
       x = "Incident Type",
       y = "Number of Denials") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```

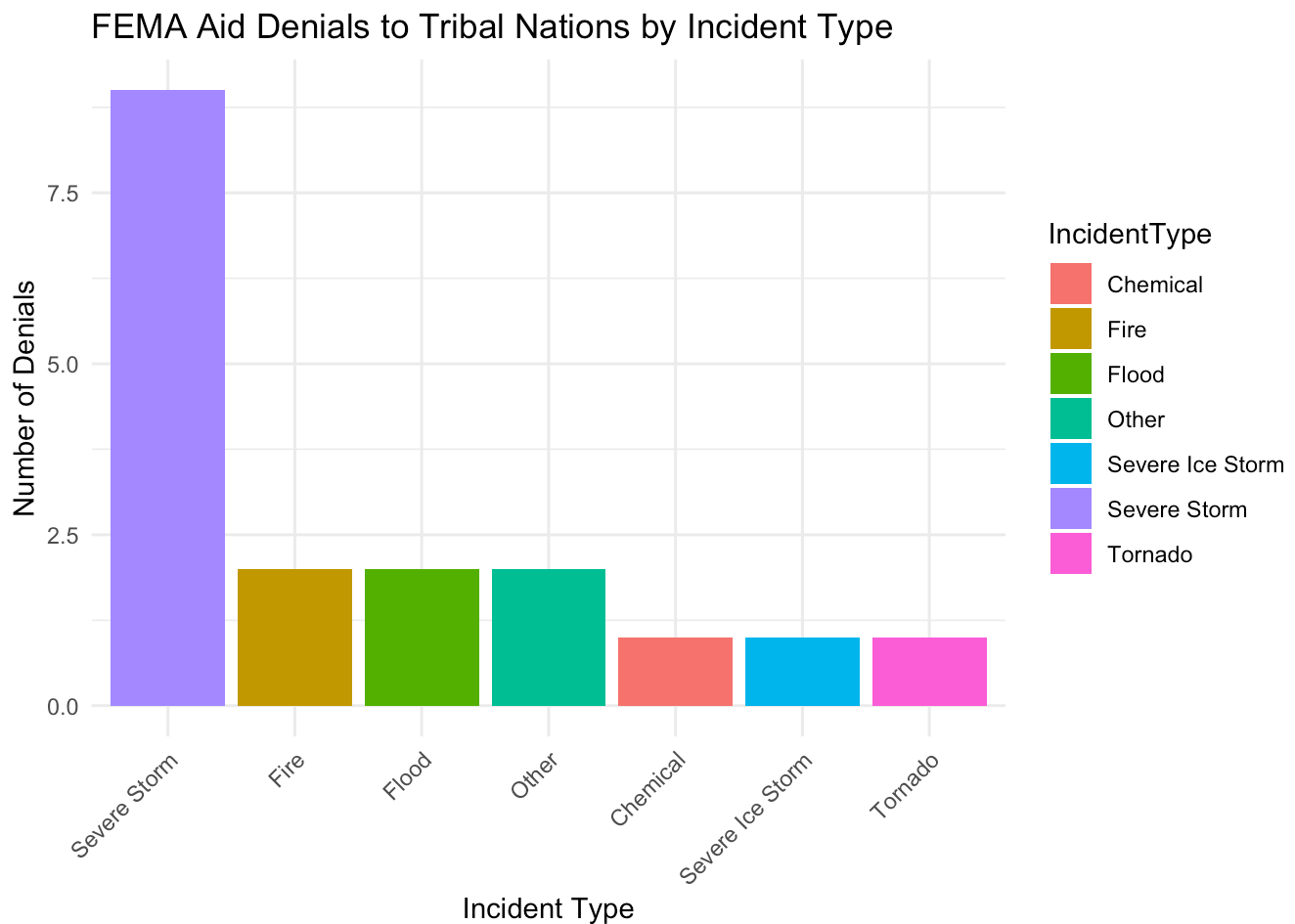


Figure 7.2. Examining the number of acceptances and denials to tribal nations over time, segmented by incident type, allows for a broader overview of why aid is being requested. Biological incidents are most frequently rewarded aid, and Severe Storm incidents are most frequently denied aid.

Examine the rate of denials per incident type, segmented by tribal nation vs. non-tribal nation, and scale results based on the total number of requests

```
# Calculate total requests by incident type and tribal status
total_requests <- data %>%
  group_by(incidentType, tribalRequest) %>%
  summarise(TotalRequests = n())
```

```
## `summarise()` has grouped output by 'incidentType'. You can override using the
## `.groups` argument.
```

```
# Filter data for denials
denials_data <- subset(data, requestResult == "Denial")

# Calculate number of denials by incident type and tribal status
denials_count <- denials_data %>%
  group_by(incidentType, tribalRequest) %>%
  summarise(Denials = n())
```

```
## `summarise()` has grouped output by 'incidentType'. You can override using the  
## `.groups` argument.
```

```
# Merge total requests and denials data  
merged_data <- merge(total_requests, denials_count,  
                     by = c("incidentType", "tribalRequest"),  
                     all.x = TRUE)  
  
# Replace NA values with 0 (for cases with no denials)  
merged_data$Denials[is.na(merged_data$Denials)] <- 0  
  
# Calculate the denial rate  
merged_data$DenialRate <- merged_data$Denials / merged_data$TotalRequests  
  
# Map tribalRequest values to labels  
merged_data$TribalStatus <- factor(merged_data$tribalRequest,  
                                   levels = c(0, 1),  
                                   labels = c("Non-Tribal", "Tribal"))  
  
# Plot the data using ggplot2  
ggplot(merged_data, aes(x = incidentType, y = DenialRate, fill = TribalStatus)) +  
  geom_bar(stat = "identity", position = "dodge") +  
  labs(title = "Scaled Denial Rate by Incident Type for Tribal vs. Non-Tribal Requests",  
       x = "Incident Type",  
       y = "Denial Rate",  
       fill = "Tribal Status") +  
  theme_minimal() +  
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```

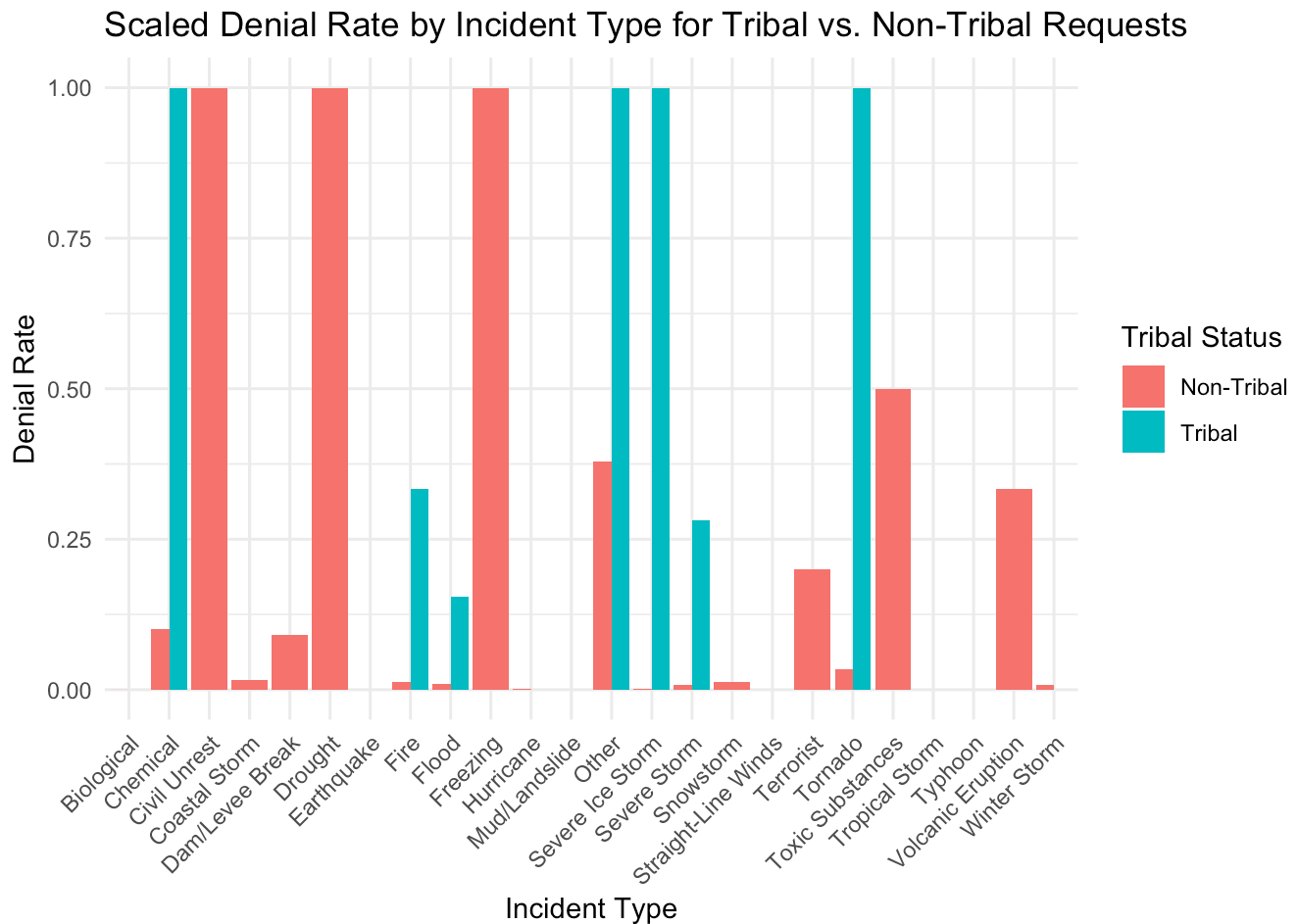


Figure 8.1

Examine the rate of acceptances per incident type, segmented by tribal nation vs. non-tribal nation, and scale results based on the total number of requests

```
# Calculate total requests by incident type and tribal status
total_requests <- data %>%
  group_by(incidentType, tribalRequest) %>%
  summarise(TotalRequests = n())
```

```
## `summarise()` has grouped output by 'incidentType'. You can override using the
## `.groups` argument.
```

```
# Filter data for acceptances
acceptances_data <- subset(data, requestResult == "Acceptance")

# Calculate number of acceptances by incident type and tribal status
acceptances_count <- acceptances_data %>%
  group_by(incidentType, tribalRequest) %>%
  summarise(Acceptances = n())
```

```
## `summarise()` has grouped output by 'incidentType'. You can override using the
## `.groups` argument.
```

```
# Merge total requests and acceptances data
merged_data <- merge(total_requests, acceptances_count,
                     by = c("incidentType", "tribalRequest"),
                     all.x = TRUE)

# Replace NA values with 0 (for cases with no acceptances)
merged_data$Acceptances[is.na(merged_data$Acceptances)] <- 0

# Calculate the acceptance rate
merged_data$AcceptanceRate <- merged_data$Acceptances / merged_data$TotalRequests

# Map tribalRequest values to labels
merged_data$TribalStatus <- factor(merged_data$tribalRequest,
                                   levels = c(0, 1),
                                   labels = c("Non-Tribal", "Tribal"))

# Plot the data using ggplot2
ggplot(merged_data, aes(x = incidentType, y = AcceptanceRate, fill = TribalStatus)) +
  geom_bar(stat = "identity", position = "dodge") +
  labs(title = "Scaled Acceptance Rate by Incident Type for Tribal vs. Non-Tribal Requests",
       x = "Incident Type",
       y = "Acceptance Rate",
       fill = "Tribal Status") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```

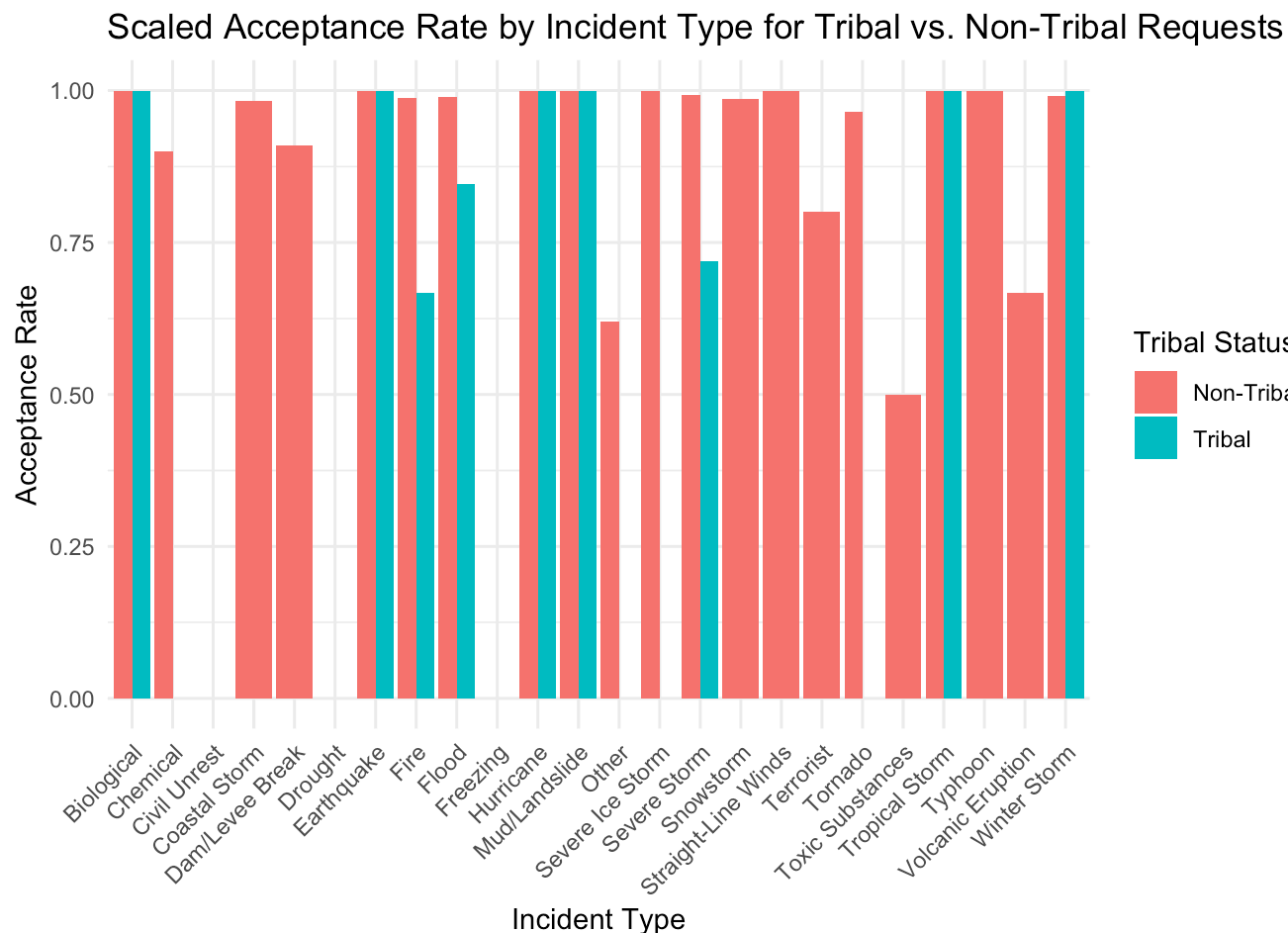


Figure 8.2. Examining the scaled denial rate by incident type for tribal vs. non-tribal requests, tribal requests are denied at a higher rate than non-tribal requests for all incident types for which both types of requests have been made.

That concludes the exploratory data analysis. Now, we will begin building a model.

BINARY LOGISTIC REGRESSION

Our goal is to analyze the likelihood that a FEMA aid request will be denied. In particular, we want to see if the coefficient for whether it was a tribal request is high and statistically significant.

First, some preliminary steps:

```
# Import additional relevant libraries
library(caret)
library(pROC)
```

```
## Type 'citation("pROC")' for a citation.
```

```
##
## Attaching package: 'pROC'
```

```
## The following objects are masked from 'package:stats':
##
## cov, smooth, var
```

```
library(dplyr)
```

```
# Clean data
# make dependent variable (request result) binary - 1 for accepted, 0 for denied
# convert into factor for classification
data$requestResult_binary <- ifelse(data$requestResult == "Acceptance",1,0)
data$requestResult_binary <- factor(data$requestResult_binary, levels = c(0, 1))
```

```
# Convert incident type to numeric by manually assigning numbers to categories
data$incidentType_numeric <- as.numeric(factor(data$incidentType,
                                              levels = c(unique(data$incidentType))))

# Convert state to numeric by manually assigning numbers to categories
data$state_numeric <- as.numeric(factor(data$state,
                                       levels = c(unique(data$state))))
```

```
# only keep potentially relevant columns
data_clean <- select(data, tribalRequest, incidentType_numeric, state_numeric, region, requestResult_binary)
head(data_clean)
```

	tribalRequest <int>	incidentType_numeric <dbl>	state_numeric <dbl>	region <int>	requestResult_binary <fct>
152	0	1	1	10	1
153	0	1	1	10	1
154	0	1	1	10	1
155	0	1	1	10	1
156	0	1	1	10	1
157	0	1	1	10	1
6 rows					

Then, balance our data and split into train/test sets.

```
table(data_clean$requestResult_binary)
```

```
##
##      0      1
## 136 24303
```

```
#note: there are 136 denials, 24226 acceptances
```

```
# filter for 0s and 1s
zeros <- data_clean[data_clean$requestResult_binary == 0,]
ones <- data_clean[data_clean$requestResult_binary == 1,]

# sample equal number of 0's and 1's
set.seed(35) # for reproducibility
sample_zeros <- zeros[sample(nrow(zeros), 136), ]
sample_ones <- ones[sample(nrow(ones), 136),]

# merge the sample datasets
data_balanced <- rbind(sample_zeros, sample_ones)

# confirm done correctly
table(data_balanced$requestResult_binary) # yes! :)
```

```
##
##    0    1
## 136 136
```

Now, split data into training/testing data.

```
set.seed(57) # for reproducibility
splitIndex <- createDataPartition(data_balanced$requestResult_binary, p = 0.7, list = FALSE)

train_data <- data_balanced[splitIndex, ]
test_data <- data_balanced[-splitIndex, ]
```

Now, we will do backward variable selection to only keep the most relevant variables in the binary logistic regression model that we train.

```
# initialize model w/ all predictors included
full_model <- glm(requestResult_binary ~ ., data = train_data, family = binomial(link="logit"))

# view model with all predictors included
summary(full_model)
```



```
##
## Call:
## glm(formula = requestResult_binary ~ ., family = binomial(link = "logit"),
##      data = train_data)
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)   -0.204704    0.564341  -0.363   0.7168
## tribalRequest  -2.642508    1.065964  -2.479   0.0132 *
## incidentType_numeric 0.024995    0.029465   0.848   0.3963
## state_numeric   0.017905    0.009738   1.839   0.0660 .
## region         -0.064056    0.070287  -0.911   0.3621
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##      Null deviance: 266.17  on 191  degrees of freedom
## Residual deviance: 246.06  on 187  degrees of freedom
## AIC: 256.06
##
## Number of Fisher Scoring iterations: 5
```

Figure 9.

```
# iteratively remove predictors until model no longer improves
backward_model <- step(full_model, direction = "backward")
```

```
## Start: AIC=256.06
## requestResult_binary ~ tribalRequest + incidentType_numeric +
##   state_numeric + region
##
##           Df Deviance   AIC
## - incidentType_numeric 1  246.79 254.79
## - region                1  246.90 254.90
## <none>                  246.06 256.06
## - state_numeric         1  249.50 257.50
## - tribalRequest         1  257.33 265.33
##
## Step: AIC=254.79
## requestResult_binary ~ tribalRequest + state_numeric + region
##
##           Df Deviance   AIC
## - region          1  247.63 253.63
## <none>             246.79 254.79
## - state_numeric   1  250.25 256.25
## - tribalRequest   1  258.63 264.63
##
## Step: AIC=253.63
## requestResult_binary ~ tribalRequest + state_numeric
##
##           Df Deviance   AIC
## <none>             247.63 253.63
## - state_numeric   1  251.77 255.77
## - tribalRequest   1  262.47 266.47
```

```
# view model
summary(backward_model)
```

```
##
## Call:
## glm(formula = requestResult_binary ~ tribalRequest + state_numeric,
##      family = binomial(link = "logit"), data = train_data)
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -0.433344   0.325481  -1.331   0.18306
## tribalRequest -2.848205   1.051010  -2.710   0.00673 **
## state_numeric  0.019251   0.009576   2.010   0.04438 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##      Null deviance: 266.17  on 191  degrees of freedom
## Residual deviance: 247.63  on 189  degrees of freedom
## AIC: 253.63
##
## Number of Fisher Scoring iterations: 5
```

Figure 10

Next steps are testing our model to see how it performs.

Note: The training data does not include all types of disasters that the testing set might, since they were split based on a randomized process. To address this, we will convert those oddball results into "Other" so that we can still use our testing data!

```
test_data_adj <- test_data
```

```
unique_test_values <- setdiff(unique(test_data_adj$incidentType_numeric), unique(train_data$incidentType_numeric))
```

```
test_data_adj$incidentType <- ifelse(test_data_adj$incidentType_numeric %in% unique_test_values, "Other", test_data_adj$incidentType_numeric)
```

```
predicted_probabilities <- predict(backward_model,
                                  newdata = test_data_adj,
                                  type = "response")
```

```
predicted_classes <- ifelse(predicted_probabilities > 0.5, 1, 0) #threshold = 0.5
```

```
cmatrix <- table(test_data_adj$requestResult_binary, predicted_classes)
print(cmatrix)
```

```
##      predicted_classes
##      0  1
## 0 22 18
## 1 17 23
```

Figure 11

MODEL TUNING AND EVALUATION

To better understand how our model's performance changes as we introduce more predictors, we will take a look at the Validation Curve, using LOOCV. This will allow us to identify if our model is overfitting or underfitting with various numbers of predictors. In addition, using LOOCV specifically will allow us to train our model more robustly, given that we have a small dataset to work with.

```
# LOOCV function:
#   traindata = training dataset
#   target = the column to predict
#   parameters = list of parameters to use in model
loocv <- function(traindata, target, parameters) {
  # for loocv, m is the size of the training data
  m = nrow(traindata)
  train_control = trainControl(method="cv", number=m)

  # train the model
  model <- train(as.formula(paste(target, parameters)),
                 data = traindata,
                 method = "glm",
                 family = binomial(link = "logit"),
                 trControl = train_control)

  return(model)
}
```

Note: When training, the code may output "Warning: There were missing values in resampled performance measures." This is most likely due to a resample where one of our result classes has 0 samples, causing the code to be unable to calculate a performance metric that involves looking at both classes.

```
# Train the model using LOOCV. Create four iterations of the model with decreasing number of predictors.
cv_model_full <- loocv(train_data, "requestResult_binary", "~ tribalRequest + state_numeric + incidentType_numeric + region")
```

```
## Warning in nominalTrainWorkflow(x = x, y = y, wts = weights, info = trainInfo,
## : There were missing values in resampled performance measures.
```

```
cv_model_three <- loocv(train_data, "requestResult_binary", "~ tribalRequest + state_numeric + region")
```

```
## Warning in nominalTrainWorkflow(x = x, y = y, wts = weights, info = trainInfo,
## : There were missing values in resampled performance measures.
```

```
cv_model_two <- loocv(train_data, "requestResult_binary", "~ tribalRequest + state_numeric")
```

```
## Warning in nominalTrainWorkflow(x = x, y = y, wts = weights, info = trainInfo,
## : There were missing values in resampled performance measures.
```

```
cv_model_one <- loocv(train_data, "requestResult_binary", "~ tribalRequest")
```

```
## Warning in nominalTrainWorkflow(x = x, y = y, wts = weights, info = trainInfo,
## : There were missing values in resampled performance measures.
```

Next we will evaluate the performance of each model iteration, collecting the training and testing accuracies for each model along the way.

```
# Create a dataframe to hold the results to plot
validCurve <- data.frame(predictors = integer(4),
                          trainAccuracy = integer(4),
                          testAccuracy = integer(4))
```

Model equation: requestResult_binary ~ tribalRequest

```
# Evaluate the model with one predictor
validCurve$predictors[1] <- 1
validCurve$trainAccuracy[1] <- cv_model_one$results$Accuracy

# Predict on testing data
test_data_one <- test_data_adj %>% dplyr::select(requestResult_binary, tribalRequest)
predicted_probabilities <- predict(cv_model_one,
                                  newdata = test_data_one,
                                  type = "prob")
predicted_classes <- ifelse(predicted_probabilities$'1' > 0.5, 1, 0) #threshold = 0.5
predicted_classes <- factor(predicted_classes, levels=c(0,1))

correct_predictions <- sum(predicted_classes == test_data_one$requestResult_binary)
total_observations <- nrow(test_data_one)
validCurve$testAccuracy[1] <- correct_predictions / total_observations
```

Model equation: requestResult_binary ~ tribalRequest + state_numeric

```
# Evaluate the model with two predictors
validCurve$predictors[2] <- 2
validCurve$trainAccuracy[2] <- cv_model_two$results$Accuracy

# Predict on testing data
test_data_two <- test_data_adj %>% dplyr::select(requestResult_binary, tribalRequest, state_numeric)
predicted_probabilities <- predict(cv_model_two,
                                  newdata = test_data_two,
                                  type = "prob")
predicted_classes <- ifelse(predicted_probabilities$'1' > 0.5, 1, 0) #threshold = 0.5
predicted_classes <- factor(predicted_classes, levels=c(0,1))

correct_predictions <- sum(predicted_classes == test_data_two$requestResult_binary)
total_observations <- nrow(test_data_two)
validCurve$testAccuracy[2] <- correct_predictions / total_observations
```

Model equation: requestResult_binary ~ tribalRequest + state_numeric + region

```
# Evaluate the model with three predictors
validCurve$predictors[3] <- 3
validCurve$trainAccuracy[3] <- cv_model_three$results$Accuracy

# Predict on testing data
test_data_three <- test_data_adj %>% dplyr::select(requestResult_binary, tribalRequest,
state_numeric, region)
predicted_probabilities <- predict(cv_model_three,
                                newdata = test_data_three,
                                type = "prob")
predicted_classes <- ifelse(predicted_probabilities$'1' > 0.5, 1, 0) #threshold = 0.5
predicted_classes <- factor(predicted_classes, levels=c(0,1))

correct_predictions <- sum(predicted_classes == test_data_three$requestResult_binary)
total_observations <- nrow(test_data_three)
validCurve$testAccuracy[3] <- correct_predictions / total_observations
```

Model equation: requestResult_binary ~ tribalRequest + state_numeric + incidentType_numeric + region

```
# Evaluate the model with all predictors
validCurve$predictors[4] <- 4
validCurve$trainAccuracy[4] <- cv_model_full$results$Accuracy

# Predict on testing data
test_data_full <- test_data_adj %>% dplyr::select(requestResult_binary, tribalRequest, s
tate_numeric, region, incidentType_numeric)
predicted_probabilities <- predict(cv_model_full,
                                newdata = test_data_full,
                                type = "prob")
predicted_classes <- ifelse(predicted_probabilities$'1' > 0.5, 1, 0) #threshold = 0.5
predicted_classes <- factor(predicted_classes, levels=c(0,1))

correct_predictions <- sum(predicted_classes == test_data_full$requestResult_binary)
total_observations <- nrow(test_data_full)
validCurve$testAccuracy[4] <- correct_predictions / total_observations
```

```
# View the results
head(validCurve)
```

	predictors <dbl>	trainAccuracy <dbl>	testAccuracy <dbl>
1	1	0.5677083	0.5500
2	2	0.6145833	0.5625
3	3	0.5885417	0.5625
4	4	0.6041667	0.5875

4 rows

```
# Plot the validation curve
ggplot(validCurve, aes(x = predictors)) +
  geom_line(aes(y = trainAccuracy, col = "Training Accuracy")) +
  geom_point(aes(y = trainAccuracy, col = "Training Accuracy")) +
  geom_line(aes(y = testAccuracy, col = 'Testing Accuracy')) +
  geom_point(aes(y = testAccuracy, col='Testing Accuracy')) +
  labs(title = "Validation Curve for Logistic Regression of FEMA Acceptance or Denial",
       x = "Model Parameters",
       y = "Accuracy") +
  theme_minimal() +
  ylim(0,1) +
  guides(fill = guide_legend(reverse=TRUE)) +
  scale_color_discrete(name = "")
```

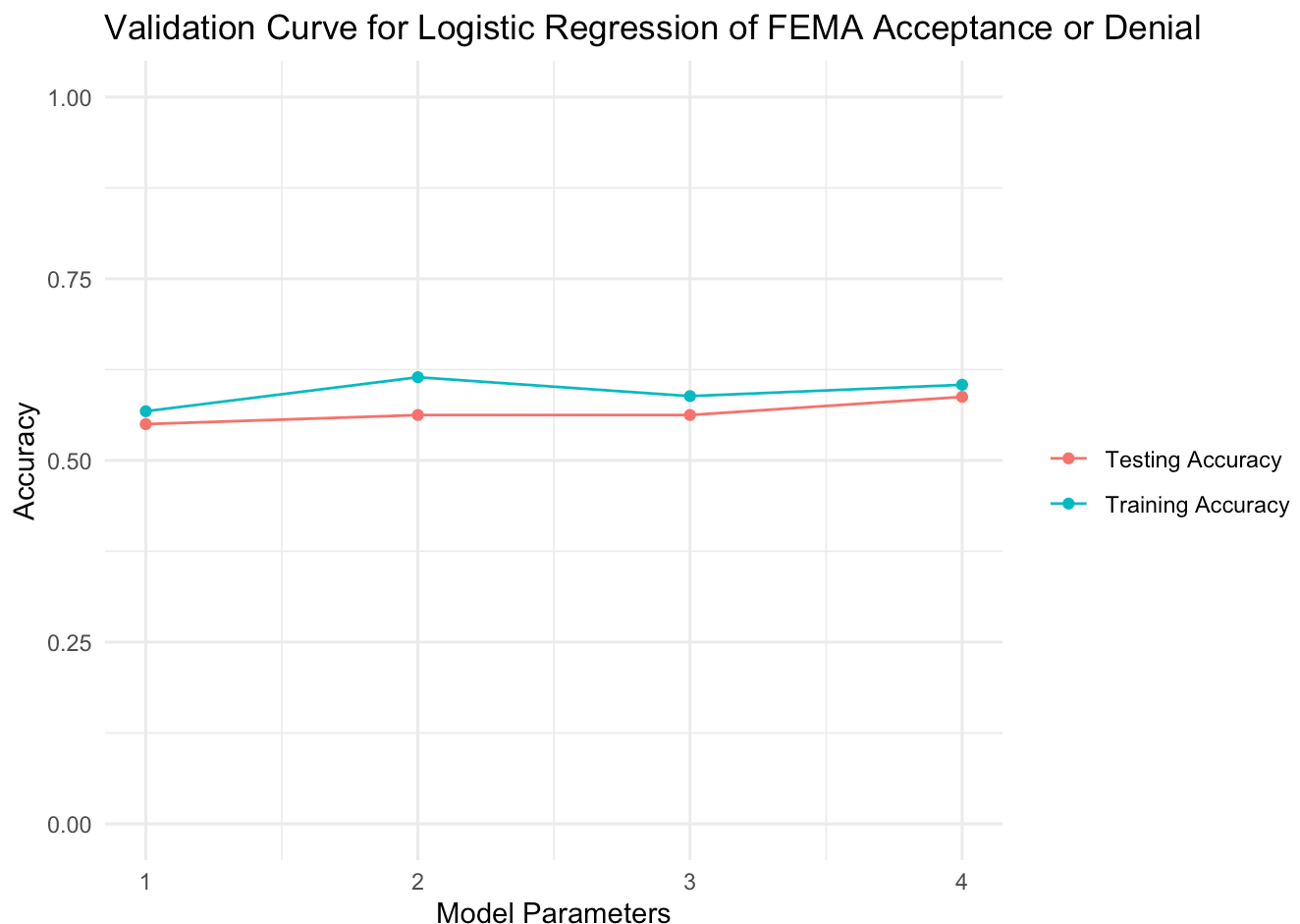


Figure 12

As we can see in the validation curve, there is not significant variation in the testing vs. training accuracies for the differing number of model parameters. We can see that the testing accuracy and training accuracy are closest in value for the model with 1 parameter and the model with 4 parameters. We will continue analyzing the model with 4 parameters to get a better sense of its overall performance.

For our final model, we will take a look at some classification metrics to evaluate its overall performance. We will calculate the metrics based on the correct classification of the accepted request class.

Here we see that the accuracy is 0.56, with 45 samples classified correctly and 35 samples classified incorrectly.

```
# Confusion Matrix
cm <- confusionMatrix(predicted_classes, test_data_full$requestResult_binary)
cm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0   1
##           0 21 14
##           1 19 26
##
##           Accuracy : 0.5875
##           95% CI : (0.4718, 0.6965)
##       No Information Rate : 0.5
##       P-Value [Acc > NIR] : 0.07282
##
##           Kappa : 0.175
##
##  Mcnemar's Test P-Value : 0.48623
##
##           Sensitivity : 0.5250
##           Specificity : 0.6500
##           Pos Pred Value : 0.6000
##           Neg Pred Value : 0.5778
##           Prevalence : 0.5000
##           Detection Rate : 0.2625
##       Detection Prevalence : 0.4375
##           Balanced Accuracy : 0.5875
##
##           'Positive' Class : 0
##
```

Figure 13

The precision is 0.56, meaning that 56% of the samples that the model classified as accepted requests were correctly identified as accepted requests. The recall is 0.55, meaning that 55% of the samples that are actually accepted requests were correctly identified as accepted requests. The F1 score, which takes into account both precision and recall, is 0.56.

```
# Overall evaluation metrics
cm$byClass
```


##	Sensitivity	Specificity	Pos Pred Value
##	0.5250000	0.6500000	0.6000000
##	Neg Pred Value	Precision	Recall
##	0.5777778	0.6000000	0.5250000
##	F1	Prevalence	Detection Rate
##	0.5600000	0.5000000	0.2625000
##	Detection Prevalence	Balanced Accuracy	
##	0.4375000	0.5875000	

Figure 14

Finally, we can see in the model summary that the tribalRequest parameter is statistically significant with a p-value of 0.0132.

```
# Look at model summary
summary(cv_model_full)
```

```
##
## Call:
## NULL
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)   -0.204704   0.564341  -0.363   0.7168
## tribalRequest  -2.642508   1.065964  -2.479   0.0132 *
## state_numeric    0.017905   0.009738   1.839   0.0660 .
## incidentType_numeric 0.024995   0.029465   0.848   0.3963
## region         -0.064056   0.070287  -0.911   0.3621
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##    Null deviance: 266.17  on 191  degrees of freedom
## Residual deviance: 246.06  on 187  degrees of freedom
## AIC: 256.06
##
## Number of Fisher Scoring iterations: 5
```

Figure 15

OVERALL LOGISTIC REGRESSION CONCLUSIONS

Overall, the metrics above mean that our model is not currently able to learn how to classify FEMA requests into acceptances or denials. While we tried to combat this with our use of LOOCV, this is most likely due to our dataset being too small to adequately train our model with. However, the finding that the tribalRequest parameter is statistically significant is important to our research question in that it indicates that whether or not a request was made by a Tribal Nation is significant to if the request is denied or accepted.

RANDOM FOREST MODEL

Given that our accuracy is fairly low from a binary logistic regression model, we will employ a random forest model to attempt to better learn how to classify FEMA requests into acceptances or denials.

```
# Load necessary libraries
library(tidyverse)
library(randomForest)
```

```
## randomForest 4.7-1.2
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:dplyr':
##
##      combine
```

```
## The following object is masked from 'package:ggplot2':
##
##      margin
```

```
library(caret)
library(ggplot2)
```

```
# Create new columns for the model
data <- data %>%
  mutate(
    requestResult_binary = factor(ifelse(requestResult == "Acceptance", 1, 0), levels =
c(0, 1)),
    tribalRequest_factor = as.factor(tribalRequest),
    incidentType_numeric = as.numeric(factor(incidentType)),
    state_numeric = as.numeric(factor(state))
  )

# Select relevant columns for the model
data_RF <- select(data, tribalRequest_factor, incidentType_numeric, state_numeric, region, requestResult_binary)

# View first few rows of Random Forest dataset
head(data_RF)
```

tribalRequest_factor <fct>	incidentType_numeric <dbl>	state_numeric <dbl>	region <int>	requestResult_binary <fct>
152 0	9	1	10	1
153 0	9	1	10	1
154 0	9	1	10	1
155 0	9	1	10	1
156 0	9	1	10	1
157 0	9	1	10	1
6 rows				

```
# Balance the dataset
# filter for 0s and 1s
zeros <- data_RF[data_RF$requestResult_binary == 0,]
ones <- data_RF[data_RF$requestResult_binary == 1,]

# sample equal number of 0's and 1's
set.seed(35) # for reproducibility
sample_zeros <- zeros[sample(nrow(zeros), 136), ]
sample_ones <- ones[sample(nrow(ones), 136),]

# merge the sample datasets
data_RF_balanced <- rbind(sample_zeros, sample_ones)

# confirm done correctly
table(data_RF_balanced$requestResult_binary) # yes! :)
```

```
##
##    0    1
## 136 136
```

```
# Split the data into training and test sets
set.seed(123)
train_index <- createDataPartition(data_RF_balanced$requestResult_binary, p = 0.7, list
= FALSE)
train_data <- data_RF_balanced[train_index, ]
test_data <- data_RF_balanced[-train_index, ]
```

```
# Train the Random Forest model
rf_model <- randomForest(requestResult_binary ~ ., data = train_data, importance = TRUE,
ntree = 500)

# Model summary
print(rf_model)
```

```
##
## Call:
## randomForest(formula = requestResult_binary ~ ., data = train_data,      importance
= TRUE, ntree = 500)
##              Type of random forest: classification
##              Number of trees: 500
## No. of variables tried at each split: 2
##
##              OOB estimate of  error rate: 22.4%
## Confusion matrix:
##      0  1 class.error
## 0 76 20   0.2083333
## 1 23 73   0.2395833
```

Figure 16

```
# Predict on the test data
predictions <- predict(rf_model, test_data)

# Confusion Matrix
confusion <- confusionMatrix(predictions, test_data$requestResult_binary)
print(confusion)
```

```
## Confusion Matrix and Statistics
##
##              Reference
## Prediction  0  1
##           0 31 15
##           1  9 25
##
##              Accuracy : 0.7
##              95% CI : (0.5872, 0.7974)
## No Information Rate : 0.5
## P-Value [Acc > NIR] : 0.0002258
##
##              Kappa : 0.4
##
## Mcnemar's Test P-Value : 0.3074342
##
##              Sensitivity : 0.7750
##              Specificity : 0.6250
##              Pos Pred Value : 0.6739
##              Neg Pred Value : 0.7353
##              Prevalence : 0.5000
##              Detection Rate : 0.3875
##              Detection Prevalence : 0.5750
##              Balanced Accuracy : 0.7000
##
##              'Positive' Class : 0
##
```

Figure 17

```
# Accuracy
accuracy <- confusion$overall['Accuracy']
cat("Accuracy:", accuracy, "\n")
```

```
## Accuracy: 0.7
```

```
# Feature Importance Visualization
importance <- as.data.frame(importance(rf_model))
importance <- importance %>% rownames_to_column(var = "Feature")
ggplot(importance, aes(x = reorder(Feature, MeanDecreaseGini), y = MeanDecreaseGini)) +
  geom_bar(stat = "identity", fill = "blue") +
  coord_flip() +
  labs(title = "Feature Importance (Mean Decrease Gini)", x = "Features", y = "Importance") +
  theme_minimal()
```

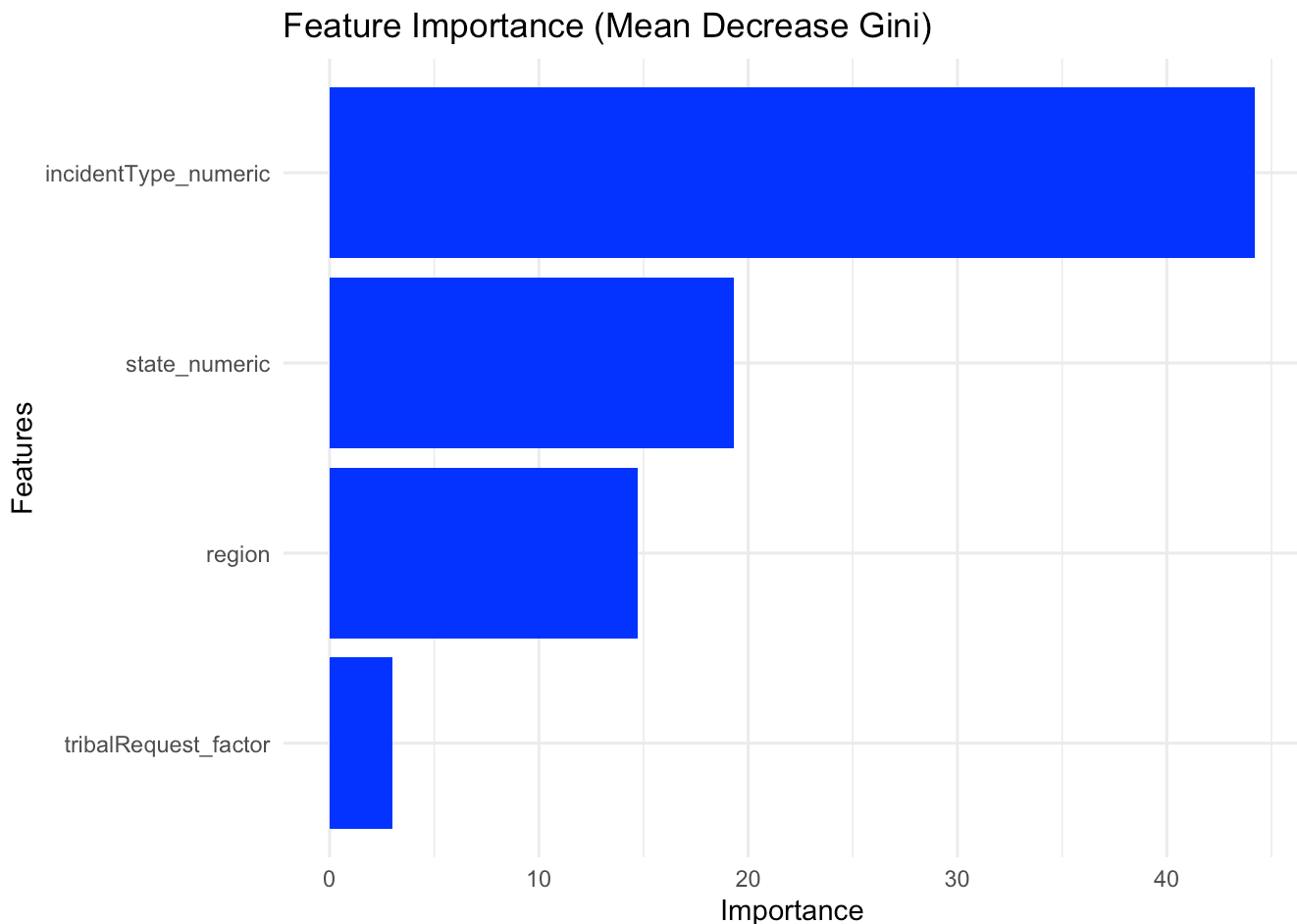


Figure 18

```
# Calculate Mean Squared Error (MSE)
mse <- mean((as.numeric(as.character(predictions)) - as.numeric(as.character(test_data$requestResult_binary)))^2)
cat("Mean Squared Error (MSE):", mse, "\n")
```

```
## Mean Squared Error (MSE): 0.3
```

```
# Plotting Error Rate Across Trees
```

```
error_rates <- data.frame(Trees = 1:rf_model$ntree,  
                          OOB_Error = rf_model$err.rate[, 1]) # OOB Error (1st column in  
err.rate)
```

```
ggplot(error_rates, aes(x = Trees, y = OOB_Error)) +  
  geom_line(color = "blue") +  
  labs(title = "OOB Error Rate Across Trees", x = "Number of Trees", y = "OOB Error Rat  
e") +  
  theme_minimal()
```

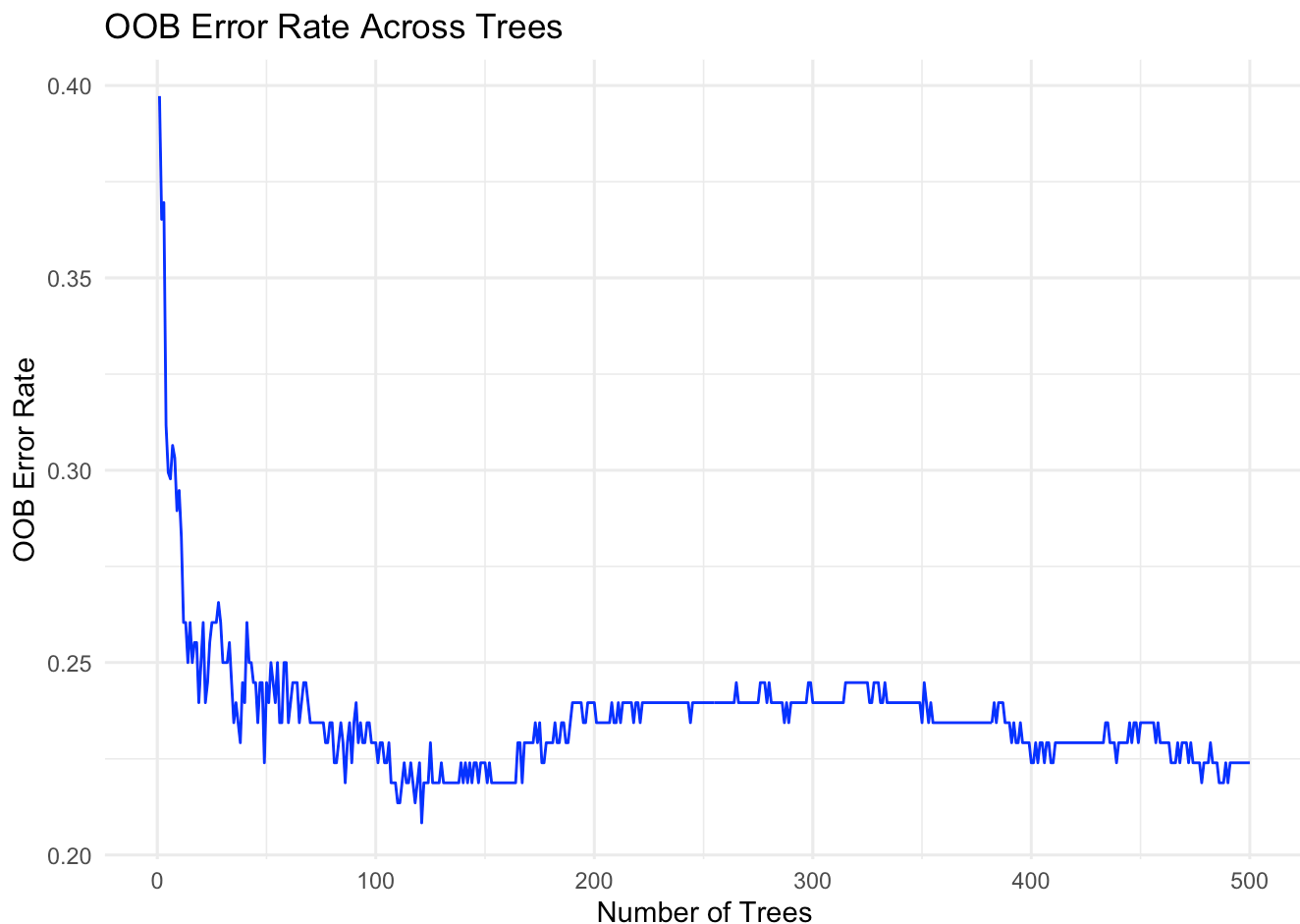


Figure 19. The optimal number of trees to reduce OOB error is around 100 trees.

```
# Train the Random Forest model with 125 trees
```

```
tuned_rf <- randomForest(requestResult_binary ~ ., data = train_data, importance = TRUE,  
ntree = 125)
```

```
# Model summary
```

```
print(tuned_rf)
```

```
##
## Call:
## randomForest(formula = requestResult_binary ~ ., data = train_data,      importance
= TRUE, ntree = 125)
##              Type of random forest: classification
##              Number of trees: 125
## No. of variables tried at each split: 2
##
##          OOB estimate of  error rate: 23.44%
## Confusion matrix:
##      0  1 class.error
## 0 76 20    0.2083333
## 1 25 71    0.2604167
```

Figure 20

```
# Predict on the test data
predictions <- predict(tuned_rf, test_data)

# Confusion Matrix
confusion <- confusionMatrix(predictions, test_data$requestResult_binary)
print(confusion)
```

```
## Confusion Matrix and Statistics
##
##              Reference
## Prediction  0  1
##           0 32 15
##           1  8 25
##
##              Accuracy : 0.7125
##              95% CI : (0.6005, 0.8082)
## No Information Rate : 0.5
## P-Value [Acc > NIR] : 9.156e-05
##
##              Kappa : 0.425
##
## Mcnemar's Test P-Value : 0.2109
##
##              Sensitivity : 0.8000
##              Specificity : 0.6250
##              Pos Pred Value : 0.6809
##              Neg Pred Value : 0.7576
##              Prevalence : 0.5000
##              Detection Rate : 0.4000
##              Detection Prevalence : 0.5875
##              Balanced Accuracy : 0.7125
##
##              'Positive' Class : 0
##
```

Figure 21

```
# Accuracy
accuracy <- confusion$overall['Accuracy']
cat("Tuned Model Accuracy:", accuracy, "\n")
```

```
## Tuned Model Accuracy: 0.7125
```

```
# Feature Importance Visualization
importance <- as.data.frame(importance(tuned_rf))
importance <- importance %>% rownames_to_column(var = "Feature")
ggplot(importance, aes(x = reorder(Feature, MeanDecreaseGini), y = MeanDecreaseGini)) +
  geom_bar(stat = "identity", fill = "blue") +
  coord_flip() +
  labs(title = "Feature Importance (Mean Decrease Gini) - Tuned Model", x = "Features",
y = "Importance") +
  theme_minimal()
```

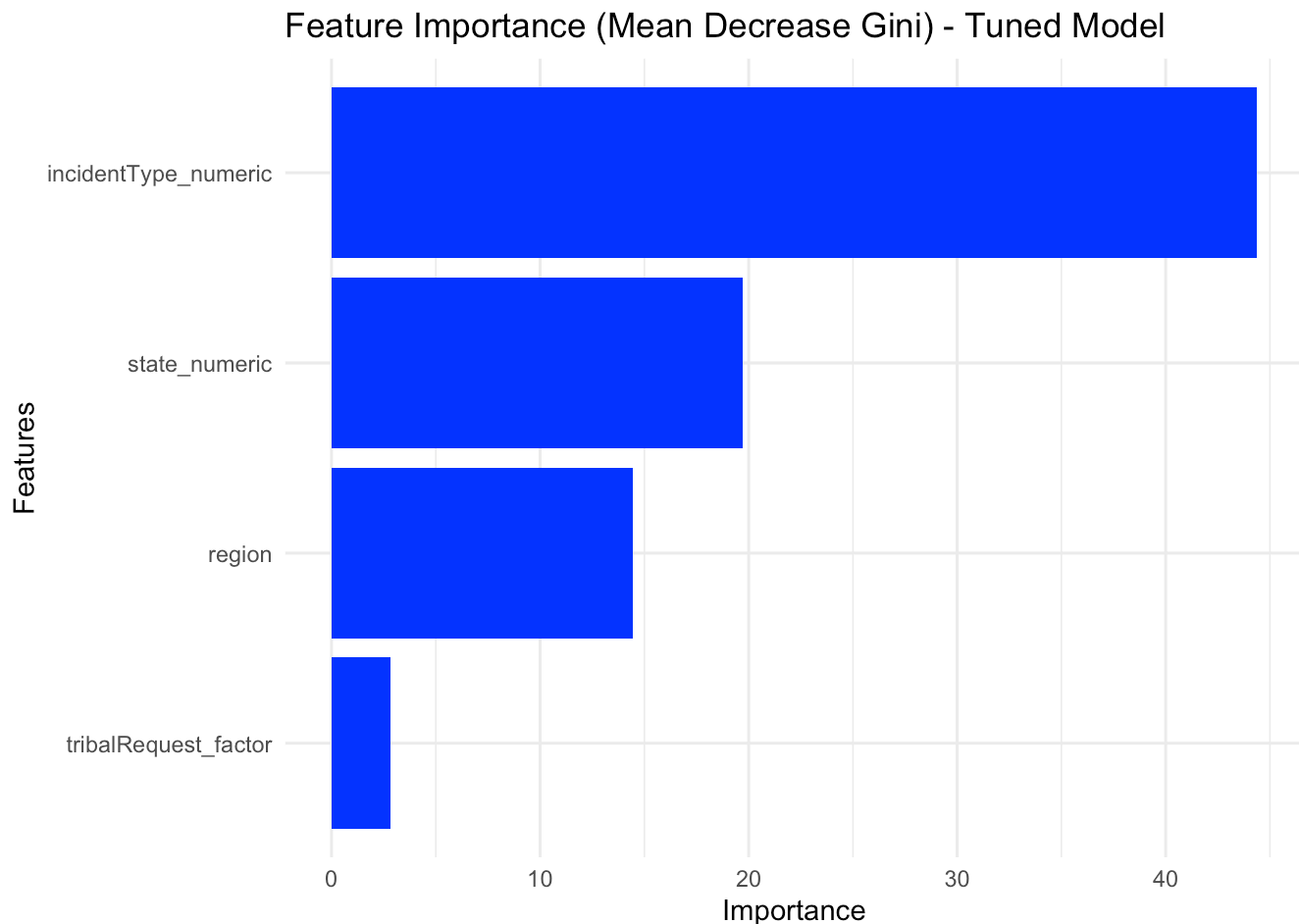


Figure 22

OVERALL RANDOM FOREST CONCLUSIONS

Overall, this model has a better accuracy than the previous logistic regression model, with a score of 0.7. The incident type was found to be the most important feature to split on within the random forest decision trees, followed by the state, region, and then if the request was a tribal request. The importance of the incident type compared to if the request was a tribal request is roughly 45 compared to 5, respectively. This result indicates that if the request was from a tribal nation was not found to be as significant of a determining factor in an acceptance or denial compared to the other features included.

The out-of-bag error rate graph shows that the error decreases from 0 to 100 trees, then increases around 200 to 350 trees, then decreases again for 400 to 500 trees. This indicates that the optimal number of trees to limit the out-of-bag error rate is around 100-125 or 500. To limit redundancy and increase efficiency, it would be best to use the smaller number of trees.

The model was rebuilt with 125 trees, and achieved an accuracy of 0.7. This result shows that the reduction in trees did not negatively affect the correct classification of the testing data, allowing us to reduce unnecessary complexity. The feature importances found for this new model were very similar to the initial model's, showing consistency in the results.