Technical Report

Nat Rubin, Kyla Hayworth, Naomi Boss

Abstract

The aim of this research project is to predict the amount of time spent in a 911 dispatcher's queue before emergency services were deployed, based on data published by the Portland Police Bureau. We combined Census data with the police report data in order to have a more holistic look at the population accessing emergency services. Our early results show that the two strongest predictors outside of priority of call and call type (both set by the dispatcher), were mean income and race. The queue time was lower for those living in a neighborhood with a higher mean income. To predict, we used a 10-fold cross validation on 6 models: multiple linear regression, ridge regression, lasso, bagged trees, random forest, and boosted forest models. For further research, we recommended a more robust data set, inclusion of spatial and temporal variables, and comparing difference between years.

Introduction

The Portland Police Bureau (PPB) serves the greater Portland, Oregon metropolitan area. Within Portland, there are 95+ neighborhoods that differ from each other greatly in terms of income, race, and percentage of population below the poverty line. We predicted that income and race would have effects on the performance of the PPB, which we assessed through dispatch time. We looked to see if we could predict response time of the PPB using income and race of the neighborhoods, as well as variables provided by the PPB that ranked the priority of the call and classified the reason for calling, such as crime or traffic complaint.

The Data

Our data is a combination of two different data sets. Census data from 2010 was taken from the City of Portland website (https://www.portlandoregon.gov/civic/56897) and data about police response time from 2012 was taken from the Portland Police Bureau (https://www.portlandoregon.gov/police/76454.). Rather than using total response time, we are using the time spent in the queue, that is, the amount of time a caller waited in the police dispatch queue before an officer was dispatched to them. The total response time includes travel time once the officer was dispatched. As there is no record of how far the officer had to travel, there was no way to standardize the total response time, so we used the time spent in the queue as a more standardized response time.

The unit of observation in this combined data set is a call to the Portland Police Bureau (PPB). The number of observations is 194,044. To predict the time in queue of the PPB we are looking at priority of the crime as rated by the PPB, racial makeup of the neighborhood (in percentage of white population), population density, and average income in the neighborhood.

Location data is majority of data that is missing in observations with missing data. This is addressed by PPB on the website. If "the incident occurred outside of the boundaries of the Portland neighborhoods or at a location that could not be assigned to a specific address in the system (e.g., Portland, near Washington Park, on the streetcar, etc.)" (PPB) there will be no data for location.

Population and economic data came from Portland Monthly: (https://www.pdxmonthly.com/articles/2016/4/1/real-estate-2016-the-city)

##Exploratory Data Analysis

Time in Queue for PPB calls

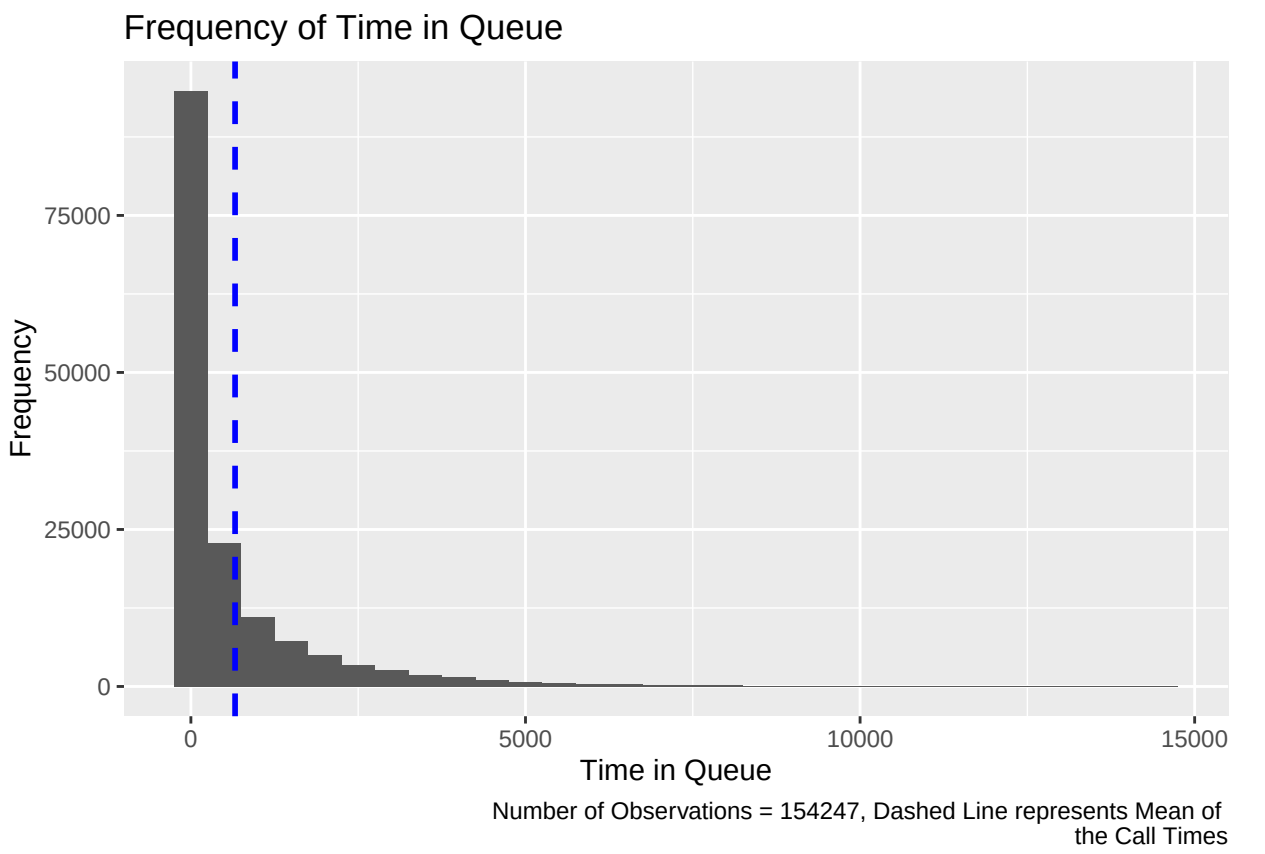
There were categorical variables in the data that coded the priority of the call and the category of the call. Priority was low, medium, or high, so we recoded the variable to be 0, 0.5, and 1, respectively. Category was recoded to be 1 for the "crime" category and 0 for all other categories, including Traffic, Disorder, and Assist. We assumed that crime calls may be given priority over other kinds of calls.

(see r chunks: Data Wrangling)

As a preliminary visualization of all of the data, a histogram was generated for time spent in the queue. This histogram is heavily skewed right, with a majority of the calls spending less time in the queue than the mean.

(see r chunks: Call Frequency)

```
ggplot(data=calls, aes(x=TimeInQueue_sec)) +
  geom_histogram(binwidth=500) +
  labs(x = "Time in Queue", y="Frequency" , title="Frequency of Time in Queue",
       caption="Number of Observations = 154247, Dashed Line represents Mean of
       the Call Times") +
  geom_vline(aes(xintercept=mean(TimeInQueue_sec)),
             color="blue", linetype="dashed", size=1)
```



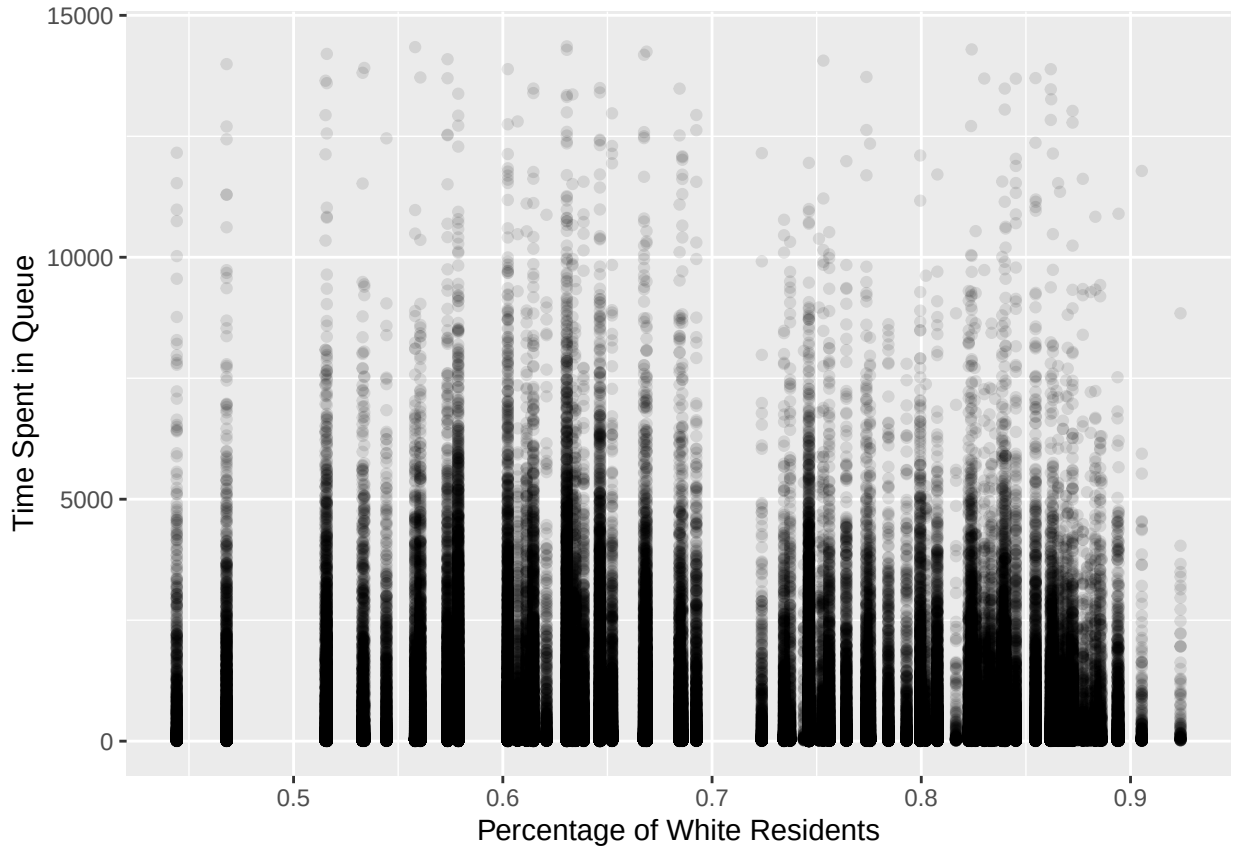


Figure 1: Frequency plot of time spent in queue by percentage of white population

```
summary(calls$TimeInQueue_sec)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     0.0    37.0   114.0   660.5   686.0 14359.0
```

Plotting bivariate correlations was our first step in exploring the data. For each, the entire data set was plotted without priority first and then also plotted with the data faceted by priority. Faceting by priority allowed us to see how important priority was to response time as well as see that generally, those with lower income spent longer times in the queue when controlling for the priority of the call.

(see r chunks: Bivariate Predictor Race, Bivariate Predictor Race and Priority, Predictor Mean Income, Predictor Mean Income and Priority, transforming variables)

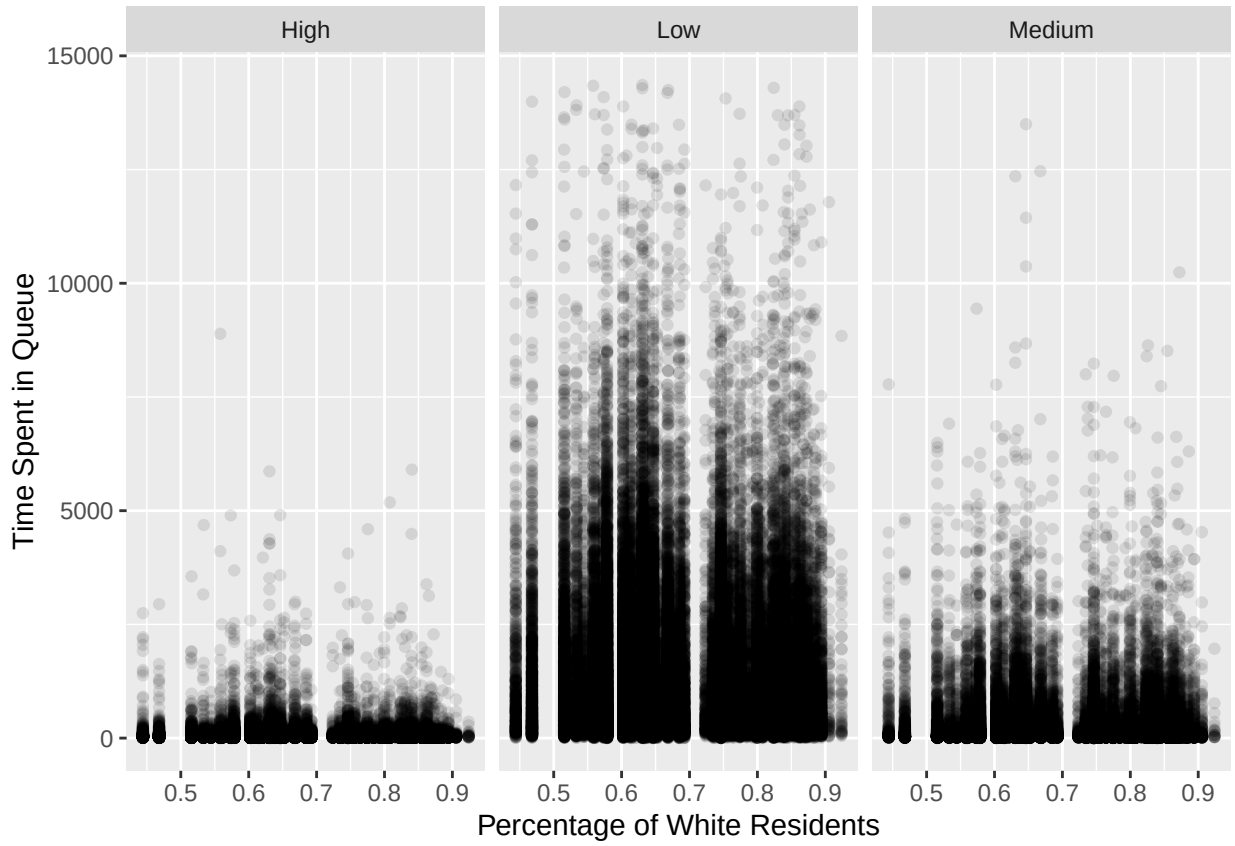


Figure 2: Frequency plot of time spent in queue by percentage of white population faceted by priority

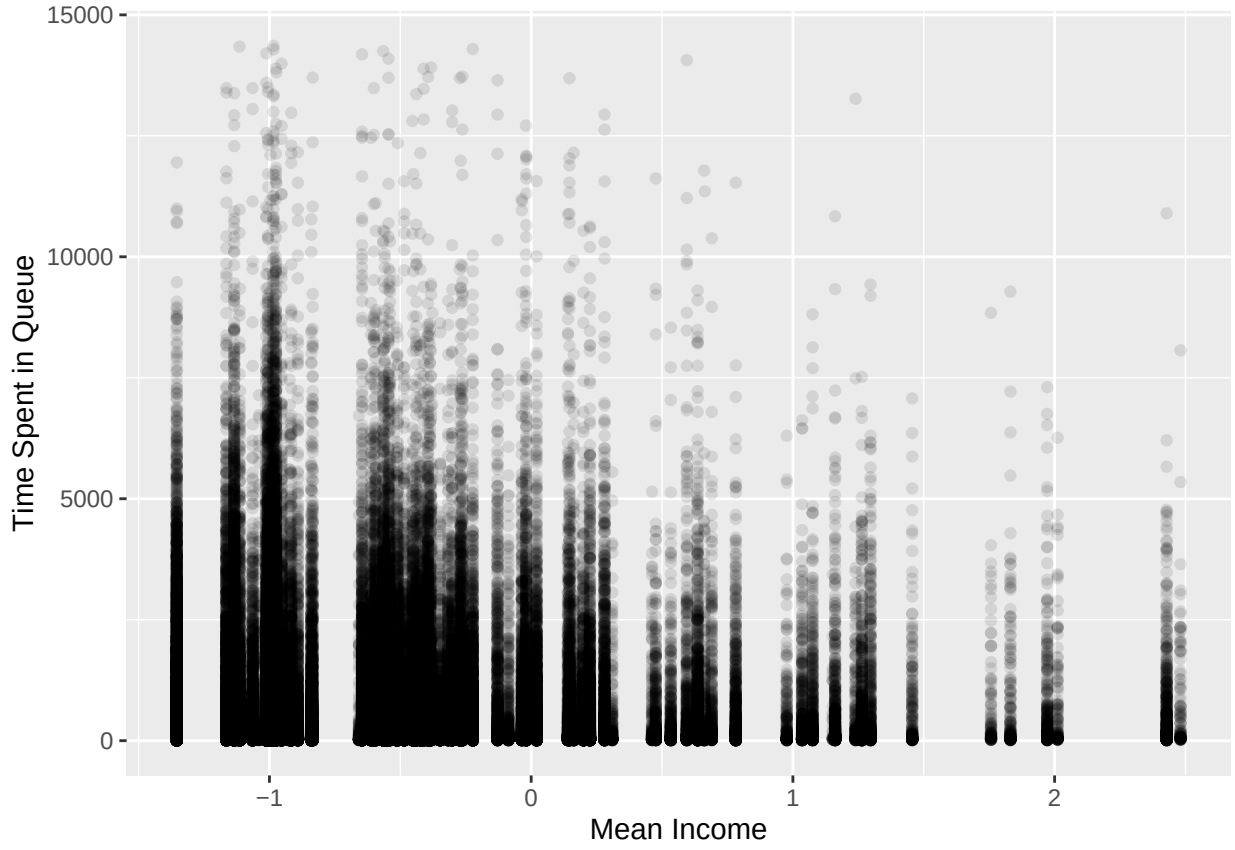


Figure 3: Frequency Plot of Time Spent in Queue Based on Mean Income

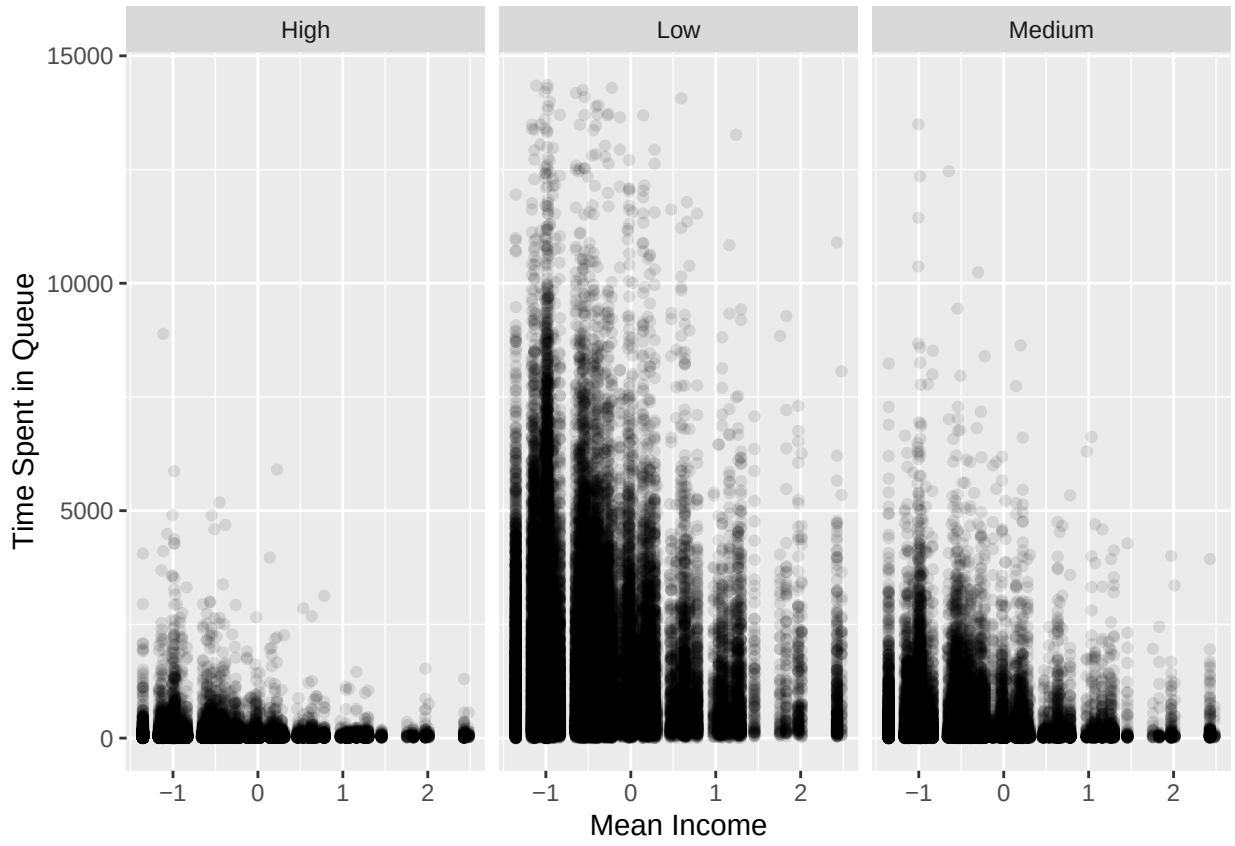


Figure 4: Frequency plot of time spent in queue based on mean income faceted by priority

Principal Component Analysis

We ran principal component analysis in an attempt of dimension reduction. Our strongest principal component only accounted for 30% of the variability, and we found that the number of principal components we needed to account for a majority of the variance was around 4. Our normal data set only utilizes 5 numerical variables, so we felt that the dimension reduction was unnecessary.

(see r chunks: PCA model building, Plotting PC1 and PC2, Scree plot)

```
## Importance of components:
##                           PC1    PC2    PC3     PC4     PC5     PC6
## Standard deviation     1.8414 1.3734 1.1103 0.99096 0.93418 0.91700
## Proportion of Variance 0.3082 0.1715 0.1121 0.08927 0.07934 0.07644
## Cumulative Proportion  0.3082 0.4797 0.5918 0.68106 0.76040 0.83684
##                           PC7     PC8     PC9    PC10      PC11
## Standard deviation     0.86061 0.77646 0.52282 0.42174 5.743e-15
## Proportion of Variance 0.06733 0.05481 0.02485 0.01617 0.000e+00
## Cumulative Proportion  0.90417 0.95898 0.98383 1.00000 1.000e+00
```

```
## List of 5
##  $ sdev    : num [1:11] 1.841 1.373 1.11 0.991 0.934 ...
##  $ rotation: num [1:11, 1:11] 0.0837 -0.4986 -0.4727 -0.2326 -0.0135 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:11] "X" "ResponseTime_sec" "TimeInQueue_sec" "TravelTime_sec" ...
##   .. ..$ : chr [1:11] "PC1" "PC2" "PC3" "PC4" ...
##  $ center  : Named num [1:11] 7.71e+04 1.10e+03 6.61e+02 4.44e+02 6.93e-01 ...
##   ..- attr(*, "names")= chr [1:11] "X" "ResponseTime_sec" "TimeInQueue_sec" "TravelTime_sec" ...
##  $ scale   : Named num [1:11] 4.45e+04 1.46e+03 1.26e+03 5.73e+02 1.14e-01 ...
##   ..- attr(*, "names")= chr [1:11] "X" "ResponseTime_sec" "TimeInQueue_sec" "TravelTime_sec" ...
##  $ x       : num [1:154234, 1:11] 1.253 1.265 -0.496 1.77 1.64 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:154234] "1" "2" "3" "4" ...
##   .. ..$ : chr [1:11] "PC1" "PC2" "PC3" "PC4" ...
##  - attr(*, "class")= chr "prcomp"
```

Modeling

We tried a basic non-cross validated linear model just to see how the skew of the variables would impact the diagnostic graphs. Upon seeing the non-random residuals and highly non-linear QQ plot, we applied log transformations to some of the variables. The linear model using the transformed variables still had problems in the diagnostic graphs, but they were improved from the non-transformed model. When we built a cross-validated linear model, we decided to use these transformed variables.

(see r chunk: linear regression, Plot of Residuals, transformed model, figs2)

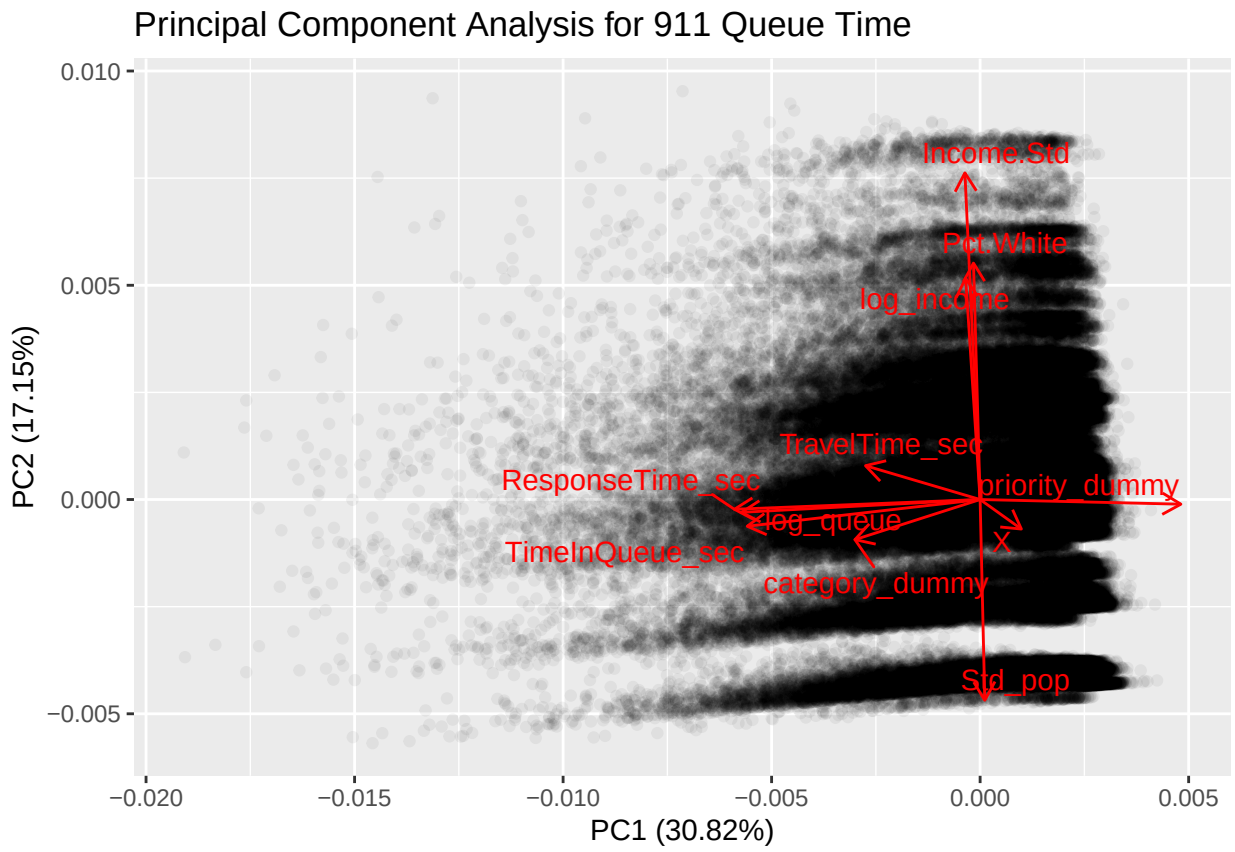


Figure 5: Plotting on first two principal component along with the bearings.

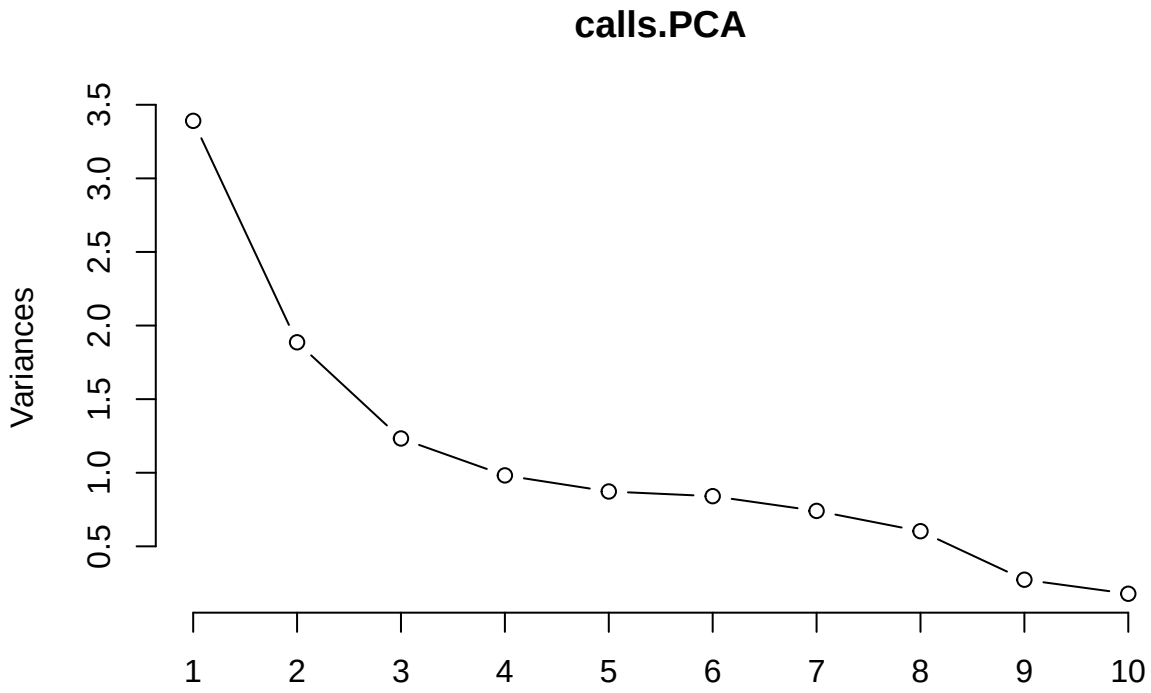
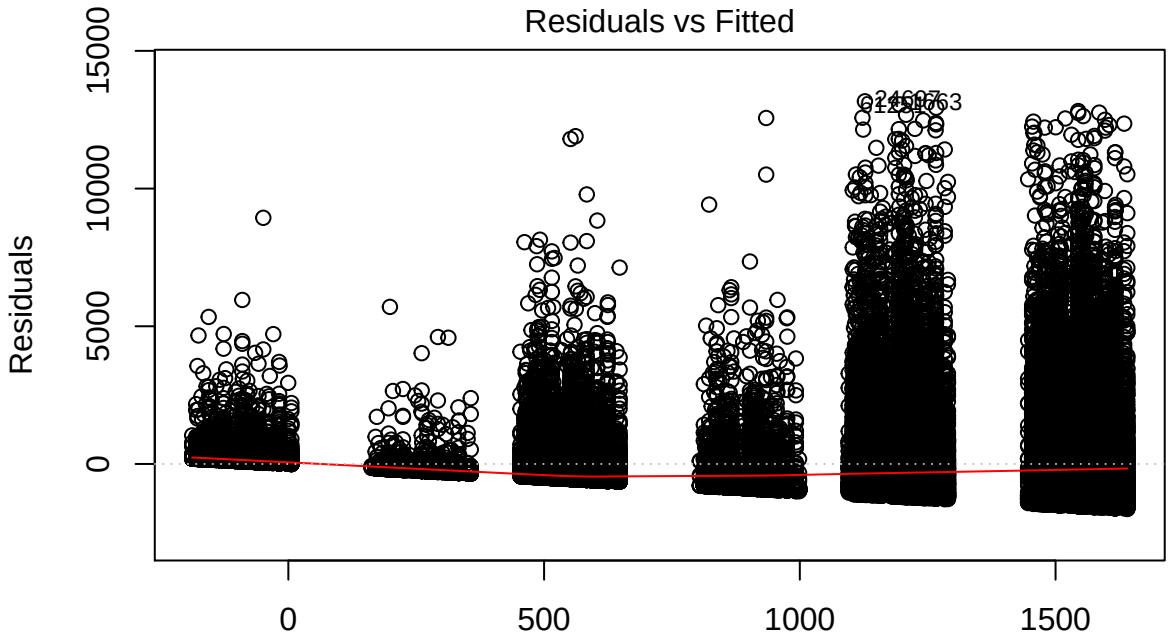
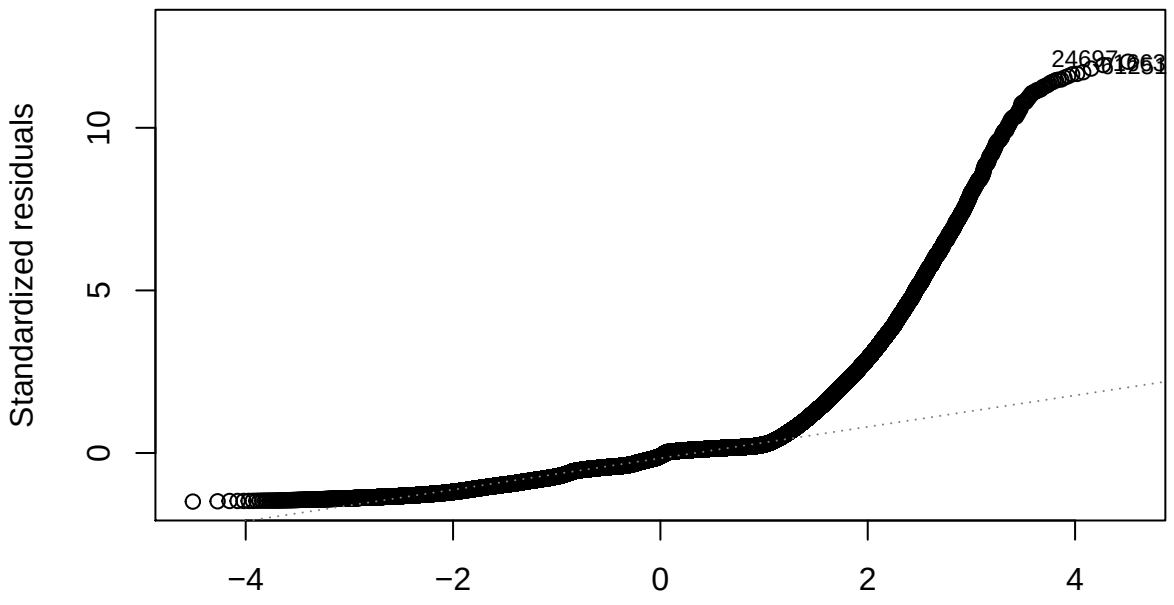
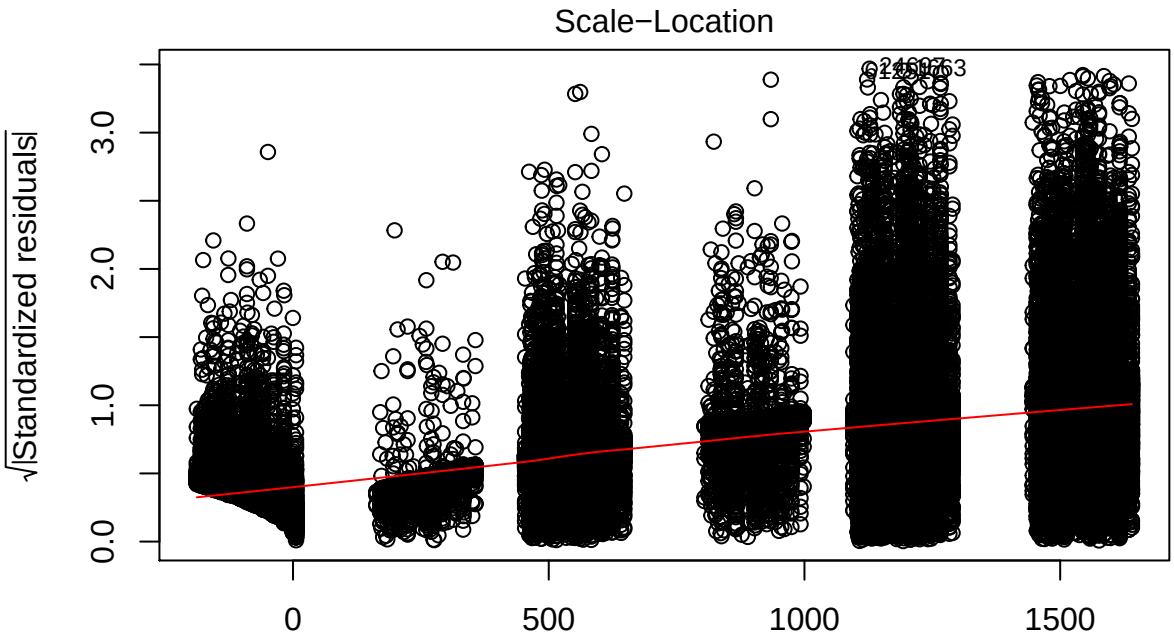
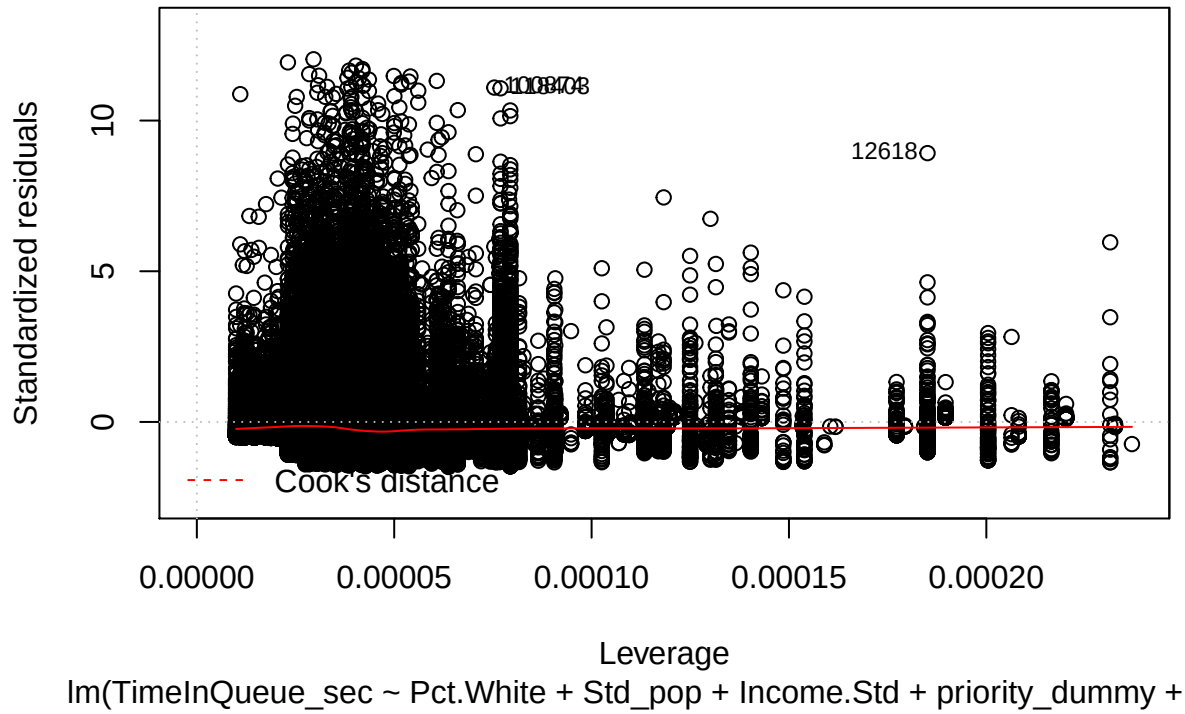


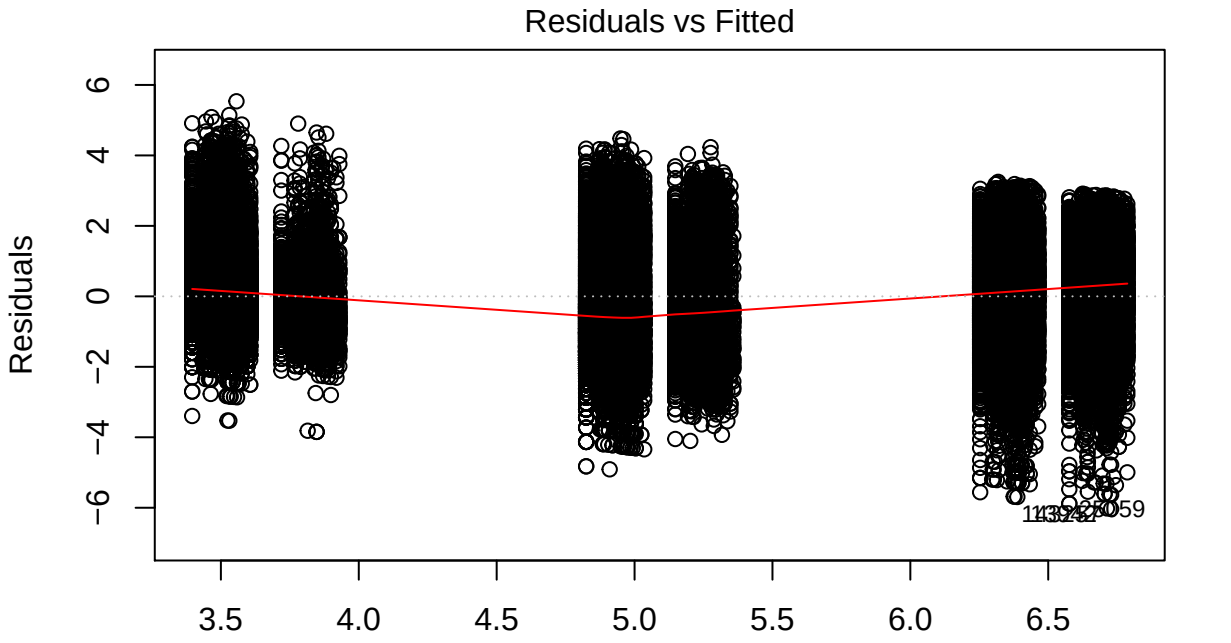
Figure 6: A screeplot to show the amount of variance accounted for by each principal component.

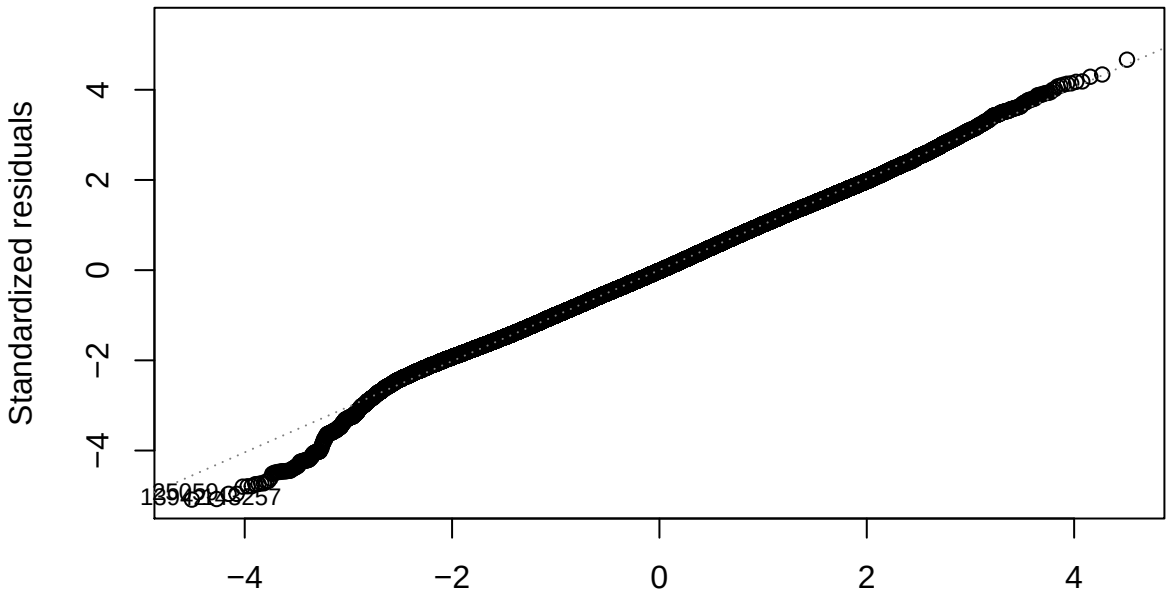

Residuals vs Fitted
Residuals
15000
10000
5000
0
0
500
1000
1500
Fitted values
lm(TimeInQueue_sec ~ Pct.White + Std_pop + Income.Std + priority_dummy + ca ..

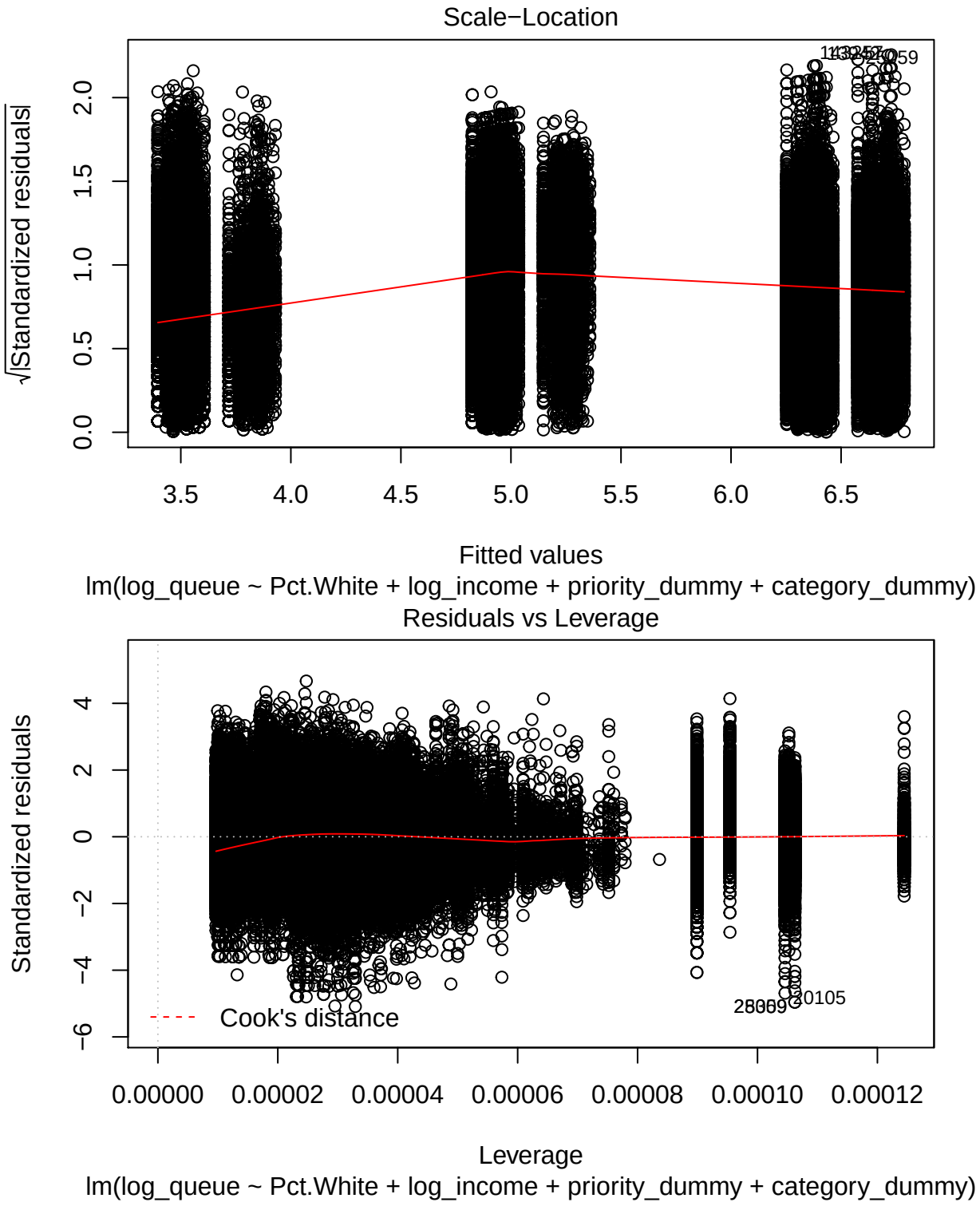

Normal Q−Q
Standardized residuals
10
5
0
−4
−2
0
2
4
Theoretical Quantiles
lm(TimeInQueue_sec ~ Pct.White + Std_pop + Income.Std + priority_dummy + ca ..


Scale−Location
√|Standardized residuals|
0.0
1.0
2.0
3.0
0
500
1000
1500
Fitted values
lm(TimeInQueue_sec ~ Pct.White + Std_pop + Income.Std + priority_dummy + ca ..


Residuals vs Leverage
Standardized residuals
0
5
10
12618
Cook's distance
0.00000
0.00005
0.00010
0.00015
0.00020
Leverage
lm(TimeInQueue_sec ~ Pct.White + Std_pop + Income.Std + priority_dummy + ca ..


Residuals vs Fitted
Residuals
6
4
2
0
−2
−4
−6
3.5
4.0
4.5
5.0
5.5
6.0
6.5
Fitted values
lm(log_queue ~ Pct.White + log_income + priority_dummy + category_dummy)


Normal Q−Q
Standardized residuals
4
2
0
−2
−4
−4
−2
0
2
4
Theoretical Quantiles
lm(log_queue ~ Pct.White + log_income + priority_dummy + category_dummy)



We tested six different models: three regression-based models and three algorithmic models. Using 10-fold cross-validation, we fit multiple linear regression, ridge regression, lasso, bagged trees, random forest, and boosted forest models. The caretEnsemble package tested the parameters for all of these models and chose the best performing model of each type. Since many of our variables were skewed, even after transformation, we anticipated the algorithmic approach to be more appropriate. However, since we wanted to engage in

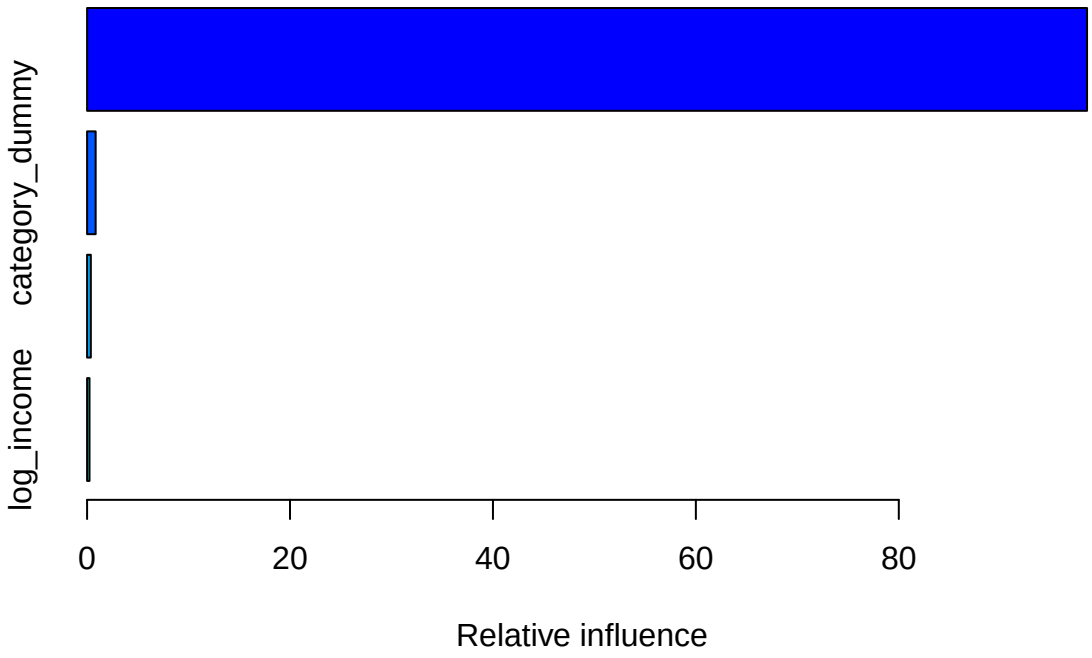


Figure 7: Summary of the Model with Least Mean Squared Error

inference regarding time in the queue, linear-based models would provide more interpretable information.

(see r chunks: caret)

Our data set was large enough that we could tune the models through 10-fold CV on training set and then test the models into a test set. The test MSEs for the best model of each type is shown below.

(see r chunks: test MSEs, test MSE summary, variable importance)

```
##     LM GLMNET RANGER TREEBAG  GBM
## 1 1.42   1.42   1.33    1.35 1.33
```

The boosted random forest model has the lowest test MSE. We used variable importance to infer how much each variable was impacting the time in the queue. Unsurprisingly, priority and category were overwhelmingly the most importance.

```
##                            var rel.inf
## priority_dummy priority_dummy  98.553
## category_dummy category_dummy   0.843
## Pct.White           Pct.White   0.367
## log_income         log_income   0.236
```

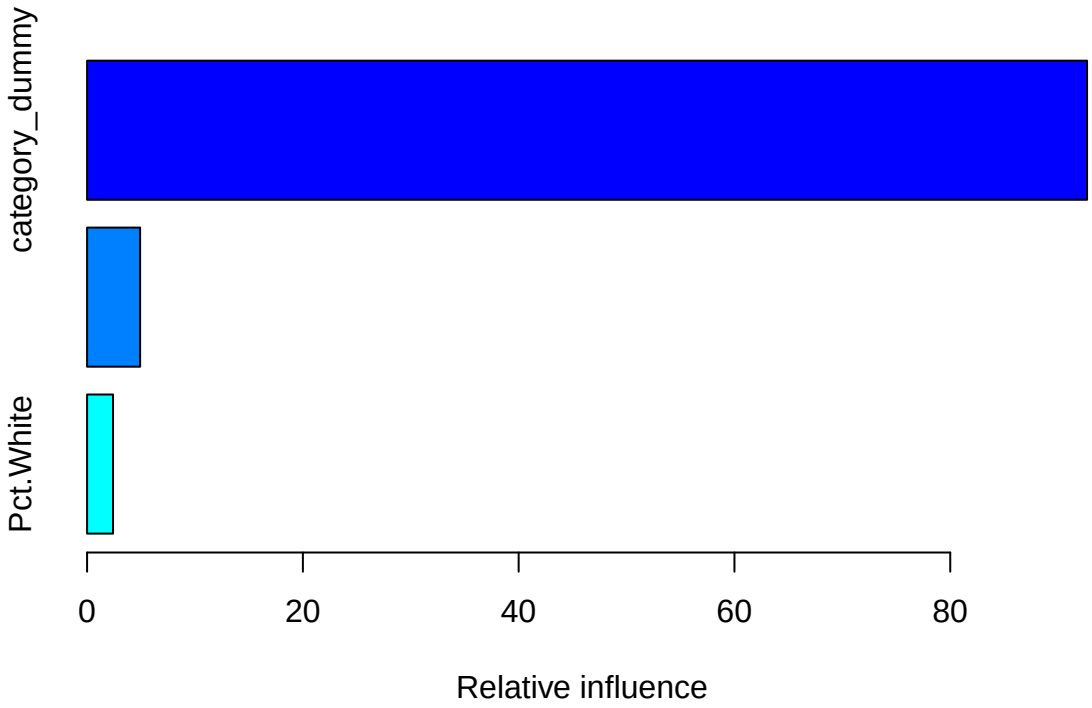


Figure 8: Summary of Boosted Random Forest Model without including priority as a predictor.

We tried a boosted random forest model without priority and then without priority and category to compare the race and income variables. Interestingly, when we removed priority, income switched places with race as the more important of the two variables, indicating that priority may have been capturing some of the importance of income in the original model.

(see r chunks: without priority, summary of no priority, without priority, summary of model without priority)

```
##                            var rel.inf
## category_dummy category_dummy   92.68
## log_income         log_income    4.92
## Pct.White           Pct.White    2.40
```

```
##                   var rel.inf
## log_income log_income    57.7
## Pct.White   Pct.White    42.3
```

Looking at our best linear regression and penalized regression models, we get more insights for inference. In the non-penalized model, all of the variables are significant at the 0.001 significance level. However, the coefficient for priority is much, much higher than the other coefficients. The penalized regression reduced the coefficients of all of the variables but did not make any of them zero, so we cannot fully discount any of the four variables.

##

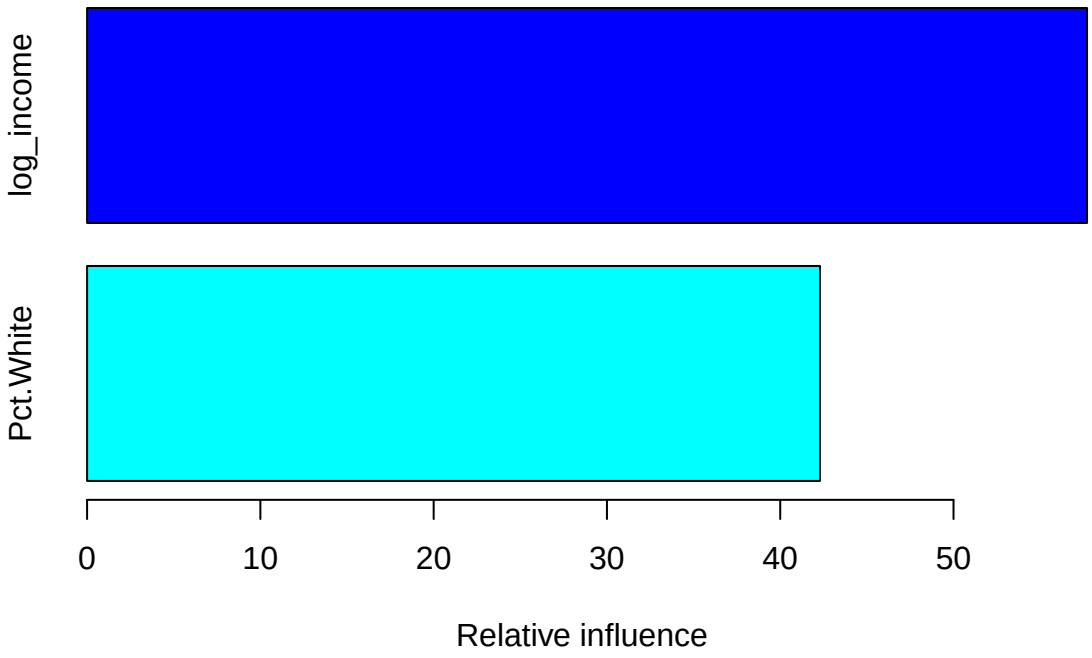


Figure 9: Summary of Boosted Random Forest Model without including priority and call type

```
## Call:
## lm(formula = .outcome ~ ., data = dat)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -6.036 -0.804 -0.017  0.811  5.535
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)     6.639560   0.020584  322.56  < 2e-16 ***
## Pct.White      -0.389505   0.028017  -13.90  < 2e-16 ***
## log_income      0.006940   0.000891    7.79  6.7e-15 ***
## category_dummy  0.325191   0.007685   42.32  < 2e-16 ***
## priority_dummy -2.855008   0.007944 -359.40  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.19 on 138805 degrees of freedom
## Multiple R-squared:  0.538,  Adjusted R-squared:  0.538
## F-statistic: 4.04e+04 on 4 and 138805 DF,  p-value: <2e-16
```

```
## 5 x 1 sparse Matrix of class "dgCMatrix"
##                       1
## (Intercept)     6.60803
## Pct.White      -0.35320
## log_income      0.00587
## category_dummy  0.32004
## priority_dummy -2.84212
```

Discussion

Unfortunately we were limited by the amount of information contained in the data. More descriptive information, like the day and time of the call, would have been very helpful in determining the volume of calls at any given time, which likely has a strong relationship to time in the queue.

##References 2010 Census Data for Portland Neighborhoods | The City of Portland, Oregon. (n.d.). Retrieved December 11, 2019, from The City Of Portland Oregon website: https://www.portlandoregon.gov/civic/56897 DeNies, R. (2016, April 1). Portland Neighborhoods by the Numbers 2016: The City. Retrieved December 11, 2019, from Portland Monthly website: https://www.pdxmonthly.com/articles/2016/4/1/real-estate-2016-the-city Dispatched Calls | The City of Portland, Oregon. (n.d.). Retrieved December 11, 2019, from The City Of Portland Oregon website: https://www.portlandoregon.gov/police/76454

R Code

We have included all of the r chunks below, in order of use in the document. In the document, the chunks will be referenced by the name included in the r chunk heading.

```
if(!require(ggfortify)){
    install.packages("ggforitfy")
}
library(ggfortify)
if(!require(glmnet)){
    install.packages("glmnet")
}
library(glmnet)
if(!require(MLmetrics)){
    install.packages("MLmetrics")
}
library(MLmetrics)
library(tree)
if(!require(caret)){
    install.packages("caret")
}
library(caret)
if(!require(randomForest)){
    install.packages("randomForest")
}
library(randomForest)
if(!require(gbm)){
    install.packages("gbm")
}
library(gbm)
if(!require(bst)){
    install.packages("bst")
}
library(bst)
if(!require(caretEnsemble)){
    install.packages("caretEnsemble")
}
library(caretEnsemble)
if(!require(tidyverse)){
    install.packages("tidyverse")
}
library(tidyverse)
if(!require(dplyr)){
    install.packages("dplyr")
}
library(dplyr)
if(!require(ggplot2)){
    install.packages("ggplot2")
}
library(ggplot2)
```

```r
calls <- read.csv("https://raw.githubusercontent.com/stat-learning/group-1/master/Data/calls.csv")

priority_dummy <- NULL
priority_dummy[calls$Priority == "Low"] = 0
priority_dummy[calls$Priority == "Medium"] = 0.5
priority_dummy[calls$Priority == "High"] = 1

category_dummy <- NULL
category_dummy[calls$FinalCallGroup == "Traffic"] = 0
category_dummy[calls$FinalCallGroup == "Disorder"] = 0
category_dummy[calls$FinalCallGroup == "Community Policing"] = 0
category_dummy[calls$FinalCallGroup == "Assist"] = 0
category_dummy[calls$FinalCallGroup == "Other"] = 0
category_dummy[calls$FinalCallGroup == "Alarm"] = 0
category_dummy[calls$FinalCallGroup == "Civil"] = 0
category_dummy[calls$FinalCallGroup == "Crime"] = 1

calls_dummy <- mutate(calls, priority_dummy)
calls_dummy <- mutate(calls_dummy, category_dummy)
```

```r
ggplot(data=calls, aes(x=TimeInQueue_sec)) +
  geom_histogram(binwidth=500) +
  labs(x = "Time in Queue", y="Frequency" , title="Frequency of Time in Queue",
       caption="Number of Observations = 154247, Dashed Line represents Mean of
       the Call Times") +
  geom_vline(aes(xintercept=mean(TimeInQueue_sec)),
             color="blue", linetype="dashed", size=1)
summary(calls$TimeInQueue_sec)
```

```r
ggplot(data = calls, mapping = aes(x = Pct.White,
                                   y = TimeInQueue_sec)) +
geom_point(alpha = 0.1)+
labs(x = "Percentage of White Residents", y = "Time Spent in Queue")
```

```r
#same as above but faceted by priority
ggplot(data = calls, mapping = aes(x = Pct.White,
                                   y = TimeInQueue_sec)) +
geom_point(alpha = 0.1) +
facet_wrap(~Priority, ncol = 3) +
labs(x = "Percentage of White Residents", y = "Time Spent in Queue")
```

```r
ggplot(data = calls, mapping = aes(x = Income.Std,
                                   y = TimeInQueue_sec)) +
geom_point(alpha = 0.1)+
labs(x = "Mean Income", y = "Time Spent in Queue")
```

```r
#same as above but faceted by priority
ggplot(data = calls, mapping = aes(x = Income.Std,
                                   y = TimeInQueue_sec)) +
geom_point(alpha = 0.1) +
facet_wrap(~Priority, ncol = 3) +
labs(x = "Mean Income", y = "Time Spent in Queue")
```

```r
income_positive <- (calls_dummy$Income.Std + 1.353877)
log_income <- log(income_positive)
hist(log_income)

log_queue <- log(calls_dummy$TimeInQueue_sec)
hist(log_queue)

calls_dummy <- add_column(calls_dummy, log_income, .after = 9)
calls_dummy <- add_column(calls_dummy, log_queue, .after = 10)

#remove negative infinity variables
calls_dummy <- calls_dummy[!is.infinite(log_queue),]
```

```r
# creating PCA model
numCalls<-calls_dummy[, -c(2:5)]
calls.PCA<-prcomp(na.omit(numCalls), center=TRUE, scale.=TRUE)
summary(calls.PCA)
str(calls.PCA)
```

```r
#plotting PCA
ggplot2::autoplot(calls.PCA,
                  alpha=0.05, loadings=TRUE, loadings.label=TRUE,
                  data=numCalls, loadings.label.repel=TRUE) +
ggtitle(label = "Principal Component Analysis for 911 Queue Time")
```

```r
screeplot(calls.PCA, type="lines")
#based on the scree plot, we should use 3 PC's.
PC1<-calls.PCA$x[,1]
PC2<-calls.PCA$x[,2]
PC3<-calls.PCA$x[,3]
```

```r
#without tranformed variables
lm_base <- lm(TimeInQueue_sec ~ Pct.White +
            Std_pop +
            Income.Std +
            priority_dummy +
            category_dummy,
          data = calls_dummy)
summary(lm_base)
```

```r
#Checking the residual graphs
plot(lm_base)
```

```r
# with transformed variables
lm_transformed <- lm(log_queue ~ Pct.White +
            log_income +
            priority_dummy +
            category_dummy,
          data = calls_dummy)
summary(lm_transformed)
```

```r
#Checking the residual graphs - these look a lot better
plot(lm_transformed)
```

```r
set.seed(23489)
train_index <- sample(1:nrow(calls_dummy), 0.9 * nrow(calls_dummy))
calls_train <- calls_dummy[train_index, ]
calls_test <- calls_dummy[-train_index, ]

preds <- c(7, 9:10, 14:15)
train_preds <- as.matrix(calls_train[,preds])
test_preds <- as.matrix(calls_test[,preds])

fit_control <- trainControl(## 10-fold CV
                            method = "cv",
                            number = 10,
                            allowParallel = TRUE)

model_list <- caretList(log_queue ~ Pct.White + log_income + category_dummy +
                          priority_dummy,
                        trControl = fit_control,
                        data=calls_train,
                        methodList = c("lm", "glmnet", "ranger",
                                       "gbm", "treebag"),
                        tuneList = NULL,
                        continue_on_fail = FALSE)

best_lm <- model_list$lm$finalModel

best_glmnet <- model_list$glmnet$finalModel

best_rf <- model_list$ranger$finalModel

best_bagged_trees <- model_list$treebag$finalModel

best_boosted_forest <- model_list$gbm$finalModel

options(digits = 3)
model_results <- data.frame(
 LM = min(model_list$lm$results$RMSE),
```

```
 GLMNET = min(model_list$glmnet$results$RMSE),
 RANGER = min(model_list$ranger$results$RMSE),
 TREEBAG = min(model_list$treebag$results$RMSE),
 GBM = min(model_list$gbm$results$RMSE)
 )
```

```
lm_yhats <- predict(model_list$lm, newdata = calls_test)
best_lm_mse <- MSE(lm_yhats, calls_test$log_queue)


glmnet_yhats <- predict(model_list$glmnet,
                        newdata = calls_test)
best_glmnet_mse <- MSE(glmnet_yhats, calls_test$log_queue)


rf_yhats <- predict(model_list$ranger, newdata = calls_test)
best_rf_mse <- MSE(rf_yhats, calls_test$log_queue)

bagged_yhats <- predict(model_list$treebag,newdata = calls_test)
best_bagged_mse <- MSE(bagged_yhats, calls_test$log_queue)

boosted_yhats <- predict(model_list$gbm, newdata = calls_test)
best_boosted_mse <- MSE(boosted_yhats, calls_test$log_queue)


options(digits = 3)
test_MSE <- data.frame(
 LM = best_lm_mse,
 GLMNET = best_glmnet_mse,
 RANGER = best_rf_mse,
 TREEBAG = best_bagged_mse,
 GBM = best_boosted_mse
 )
```

```
print(test_MSE)
```

```
summary(best_boosted_forest)
```

```
no_priority <- caretList(log_queue ~ Pct.White + log_income + category_dummy,
                        trControl = fit_control,
                        data=calls_train,
                        methodList = c("gbm"),
                        tuneList = NULL,
                        continue_on_fail = FALSE)
```

```
summary(no_priority$gbm$finalModel)
```

```r
no_p_or_c <- caretList(log_queue ~ Pct.White + log_income,
                       trControl = fit_control,
                       data=calls_train,
                       methodList = c("gbm"),
                       tuneList = NULL,
                       continue_on_fail = FALSE)
```

```r
summary(no_p_or_c$gbm$finalModel)
```

```r
summary(best_lm)

coef(best_glmnet, 0.00254)
```