Matt Alvarez-Nissen and Lilla Petruska
MS&E 226
10/27/2020

MS&E 226 Project Part 1 - Neighborhood Outages

Pacific Gas & Electric (PG&E) has made headlines over the last few years, largely due to blame over faulty equipment deemed responsible for sparking California's worst wildfires in that time. As a result, the utility company has imposed "public safety power shutoffs" in the wake of dangerous fire conditions to reduce the possibility of their equipment contributing to any new wildfires. It is also common for communities to experience power shutoffs during distant wildfires or shortly after they start as they might prevent their spread. For our project, we investigate these PG&E outages and what factors influence their scope (temporally and spatially). Specifically, are there factors in communities that impact PG&E's service to its customers and response to such power outages?

Investigating the data

By combining data on PG&E outages with demographic information, we hope to answer some questions about which characteristics within Census tracts are associated with the duration and extent of power outages. Are there certain racial, socioeconomic, or educational attainment indicators within communities that influence PG&E's response time to outages? Does PG&E's performance show neglect for its customers in certain Census tracts? Is median duration per tract higher if there is a higher proportion of non-white residents? Given the contentious nature of PG&E's performance over the last few years, it is especially relevant now to understand how their electricity system infrastructure impacts Californians and it is imperative that we focus this investigation through the lens of equity. Predicting characteristics about PG&E's outage will be increasingly important in the future as wildfires become more frequent and severe. Understanding what impacts outage duration and extent can give us insights into which populations might be vulnerable to being left without power for longer periods of time and thus inform decisions about where energy infrastructure may need to be updated and strengthened in preparation for times of crises.

Our dataset combines data from three sources: the United States Census Bureau's 2018 American Community Survey, the 2010 Decennial Census urban-rural classification of Census tracts, and PG&E electricity outage data for the last year (found here). The first provides demographic data at the Census tract level in California, including information on race, population density, income level, educational attainment, tenure (homeowner vs renter), and home vacancy rates. The second gives information about urban versus rural land use in tracts. The third was scraped from PG&E directly by Simon Willison and contains information on outage duration, location, and customers affected. By georeferencing each outage, we were able to calculate a set of summary statistics of PG&E outages for each Census tract that is within PG&E's service area. Our dataset for analysis is the result of joining these three separate datasets, giving us demographic, land use, and outage data for 2722 Census tracts in California.

We have a few concerns about the quality of our data, including that the location data PG&E provides is only a single coordinate for each outage, and thus there is not reliable information concerning the extent of the entire affected outage area. Additionally, PG&E does not provide a data dictionary, so we must infer what the different variables mean based on their names. Another issue in the structure of the data is that While PG&E serves a majority of Northern California, the geography has an incredibly diverse demographic and environmental makeup. Our random train-test split hopefully accounted for variation across the Census tracts, but there is a possibility that the tracts within each of these groups share similar characteristics. Finally, our data had relatively few observations, as we only consider the Census tracts with outages. We deemed this, as opposed to including all Census tracts in California, the more appropriate subset to analyze. Given that PG&E's service area is restricted, we felt including all Census tracts would unfairly weight the `0` observations for our outcome variables. We realize, however, that this must introduce some selection bias. If allowed more time, we would also deploy our models on the entirety of PG&E's service area (which was not immediately available).

Data processing was a three step process, requiring Census data cleaning, PG&E data cleaning, and merging. 2018 ACS measurements for race, population density, income (as percentage of Area Median

Table 1: Very High Correlations Amongst Covariates

covariate_1	covariate_2	correlation
prop_less_than_hs	prop_college	-0.7904132
prop_high_school	prop_college	-0.7850177
prop_college	prop_latino	-0.7607331
prop_vli	prop_hi	-0.7408347
prop_li	prop_hi	-0.7171673
prop_mi	prop_hi	-0.6743541
prop_less_than_hs	prop_white	-0.6725149
prop_latino	prop_white	-0.6717423
prop_renter	prop_hi	-0.6274611
n_outages_sq_km	pop_density_sq_km	0.6993085

Income) level, educational attainment, tenure (homeowner vs renter), and home vacancy rates were calculated as proportions. Additionally, we used 2010 Decennial Census (the most recent data available) to determine the proportion of the population classified as urban or rural. We removed one covariate from each of category to reduce issues with collinearity in the regression model. The `tigris` and `tidycensus` packages were used to gather and clean all Census data. Outage data was taken from Simon Willison's PG&E scraper, and was filtered to only include outages that were over. There is about a year of data, from October 2019 - October 2020. Mean customers affected was derived by taking the average of `min_estCustAffected` and `max_estCustAffected`, and `possible_duration_hours` was used as a measure of outage duration in hours. This data was georeferenced to Census tracts using latitude and longitude, with help from the `tigris` and `sf` packages. ACS and outage data were then merged by grouping outage data by tract, after determining median outage duration, median customers affected, and total number of outages by tract. A binary flag was generated to note if a tract had above median customers affected by outages. We also generated a density statistic for outages per square km. Full data cleaning scripts are available in Appendix C.

Our continuous response variable is the median outage duration (hours) in each Census tract. We chose this variable because it may serve as a proxy for PG&E service quality (how fast PG&E can bring power back on for communities). Our binary response variable determines if the number of customers affected by outages within a Census tract is above the median. We chose this because it may help us understand what kind of neighborhoods are more heavily impacted by these power outages.

After exploring our data, it became clear to us that correlations between covariates would be a major issue to consider, especially given the high correlation between variables like education and race. For the regression model, we thus decided to include interaction terms in our data to capture this relationship. We also included log transformations on variables following a chi-squared distribution, including proportion Latino and proportion with less than a high school education. There were also a handful (about 15) Census tracts that contained `NA` values, and most of those with `NA`s did not have other useful information so we decided to drop them. Finally, our EDA let us see some interesting patterns concerning race, education, and tenure especially. As median outage duration increased, the proportion of white residents, college-educated residents, and owners decreased. These patterns demonstrated to us that our data had at least some worthwhile predictive power. These plots are included in Appendix D.

Table 2: Prediction Error, R^2, and CVerror for Regression Models

model	RMSE	Rsquare	CVerror	mean(median_outage_duration_hr)
transform	4.030581	0.0388899	4.065080	NA
all interactions	3.856560	0.1200903	4.252742	NA
transform + interactions	3.850536	0.1228373	4.235039	NA
selected transform + interactions	4.012492	0.0474975	4.100377	NA
ridge	4.041137	0.0338489	4.110233	NA
lasso	4.031624	0.0383926	4.111318	NA

Predictions

Here we will discuss the general modeling process and decisions made in creating both regression and classification models. The full process and code can be found in Appendix B.

Our regression model was built to predict our continuous outcome variable, median outage duration (hours), in each tract. Each model we created was trained on a training set comprising 80% of our data. A prediction error and R^2 value were calculated for each model to evaluate how well each fit to the training set. To determine which model was the best, we used K-fold Cross Validation as an estimate of prediction error on the test set. Results can be found in Table 2.

For a baseline model, we used a basic OLS linear regression, as it will minimize the sum of the squared differences between the observed values and the corresponding fitted values. This resulted in a RMSE of 4.032, R^2 value of 0.038. This was not surprising, as our dataset is relatively small and all our covariates, while still important environmental indicators, are from a separate data source. Given that our baseline model had a high RMSE and low R^2 value, we applied a series of modifications to our model. Our GGPairs plot of covariates (see Appendix D) showed that two covariates, 'prop_latino' and 'prop_less_than_hs' had heavy-tailed distributions and thus could benefit from log transformations. We transformed the data and ran the baseline model again with transformed covariates. This had a small improvement in RMSE and R^2. Knowing that collinearity may be an issue because our covariates were measured as proportions of a series of variables, we also decided to create a model including all interaction terms. This improved RMSE and R^2 to 3.857 and 0.12, respectively, though was expected because the large number of additional covariate terms can result in overfitting of the data. To control for this, we tried only including interaction terms for covariate that had a absolute correlation coefficient of 0.5. This model had an slightly higher RMSE and lower R^squared, but likely overfit the data less than before. Finally, we used ridge and lasso regression techniques to select for the most meaningful covariates. Both these methods had worse results than the baseline, as the baseline likely overfit the data relatively more than the regularization models. Overall, all our models had poor performance which indicates that our covariates have low predictive power. While we had hoped the demographic and land use data might help us understand PG&E's performance, there are likely other factors we must consider.

Alongside calculating RMSE and R^squared to understand the performance of our models on the training set, K-fold Cross Validation was done to estimate generalization error. While also testing our models on a validation set would have produced an unbiased estimate of generalization error, lower error results could have happened by luck and underestimated the true generalization error. Our K-fold Cross Validation results produced biased estimates of generalization error because the cross validation was done on the same training dataset. However, each validation was done on subset of our training data. We accept that using CVerror to estimate generalization error will be biased and we will not know the true generalization error until testing our models on the test set.

When looking at Table 2, we see that the baseline, transform, and selected transform + interaction models have the lowest CVerrors. The simple transform model performs the best across the board, likely because the log transformations allow us to pick up the effects of small changes in the 'prop_latino' and 'prop_less_than_hs' variables on the outcome variable without overfitting. Overall, all models performed relatively poorly and did not improve much compared to the baseline model. Any improvements are highly optimistic and lower CVerror could be simply attributed to chance. This is partly because the baseline itself was a poor model and more complex models did nothing more than overfit to the training data. Using K-fold Cross Validation

Table 3: Above Median Customer Affected Confusion Matrix Metrics

metric	base	final
Mean 0-1 Loss	0.1927878	0.0088275
Precision	0.6444444	0.6307692
Sensitivity	0.0674419	0.0953488
Specificity	0.9907675	0.9861512
Type I Error Rate	0.0092325	0.0138488
Type II Error Rate	0.9325581	0.9046512
False Discovery Rate	0.3555556	0.3692308

as our evaluation of performance, the transform model performed the best and had an estimate for prediction error of 4.066. This model does not overfit the test data as much as the more complex ones, while accounts for the heavy-tailed nature of its covariates.

If we take a step back and look at the sample mean of our training data as an even simpler baseline, the average median outage duration across census tracts is 2.34748 hours. When comparing our model performance to this, each model would still perform poorly, predicting a sample mean of potentially 4 hours greater than the real value (technically it could only predict ~2.3 hours lower because time cannot be negative!). When considering the policy implications of this, predicting a power outage that is substantially longer could have negative impacts on communities by allocating resources poorly. However, preparing for longer outages may be smarter in reality given the unpredictable nature of wildfires and natural disasters. It is always better to be safer than sorry.

In classification, our major goal was to identify as many true positives as possible, because we wanted a model optimized for proactive decision-making. In other words, we were fine with relatively higher false positives, because the model is designed to give warning to particular neighborhood types if they are at higher risk of having high numbers of customers affected by outages. As such our major goals were to reduce mean 0-1 loss and increase sensitivity.

For a baseline, we used a logistic regression of the model including all covariates. This performed fairly well, with a mean 0-1 loss of about .193 and sensitivity of .067. However, given the issues of collinearity pointed out before, we wanted to see if we could develop a better model that reduced these issues. Thus, we used Principal Components Analysis (PCA) to reduce the covariates, and chose to include the first 7 components (which explained about 70% of the variance). Next, we used K-nearest neighbors (KNN) on those first 7 components, and did a 10-fold cross-validation to derive the optimal K (which was 37). Using confusion matrices, we then produced a comparative table between the two models (shown in Table 3). Mean 0-1 loss greatly improved, now about .009 while sensitivity increased to about .095. Obviously other metrics took a hit, including AUC, specificity, and precision. For the purposes of our model though, this was deemed acceptable.

Overall, there is reason to believe that our model is overly optimistic and biased. While CV was employed to reduce this possibility, the relatively low explanatory power of our covariates make this more likely. Additionally, the number of components chosen was relatively arbitrary. 7 components occurred after the break in the scree plots, but was chosen because it explained an acceptable amount the variance. KNN might have also been a suboptimal classification approach. However, given the constraints of our data, we were content with the KNN model. We felt it made use of the highest explanatory power of the data without overfitting to noise excessively. Hopefully this remains true when we deploy it on the test set.

Appendix A - Report Setup

This appendix provides the code used to setup up this report in RMarkdown.

```
# Please exclude this page from page count
# Libraries
library(tidyverse)
library(Hmisc)
library(data.table)
library(foreach)
library(skimr)
library(knitr)
library(kableExtra)
library(GGally)
library(caret)
library(pROC)
library(here)
library(glmnet)
library(corrr)
library(elasticnet)
library(cvTools)

# Load Data
acs_outages <- read_csv(paste0(here::here(), "/cleaned_data/acs_outages_train.csv"))

# Parameters
lambdas <- 10^seq(3, -2, by = -.1)
## List of outcome variables
outcome_vars <- c("median_outage_duration_hr", "above_median_cust_affected")

## List of covariates
continuous_vars <-
  names(acs_outages %>% select(-c(all_of(outcome_vars), GEOID)))

# Functions
eval_results <- function(model, true, predicted) {
  model <- enquo(model)
  SSE <- sum((predicted - true)^2)
  SST <- sum((true - mean(true))^2)
  R_square <- 1 - SSE / SST
  RMSE = sqrt(SSE/length(true))

  # Model performance metrics
  data.frame(
    model = as_label(model),
    RMSE = RMSE,
    Rsquare = R_square
  )
}
```

```
# Regression df
reg_train <-
  acs_outages %>%
  select(-c(GEOID, above_median_cust_affected, prop_white, prop_eli, prop_college, prop_owner, rental_va
```

```r
  drop_na()

# Classification df
# Set up data for classification (and drop NAs)
acs_outages_class <-
  acs_outages %>%
  select(
    -c(GEOID, median_outage_duration_hr)
  ) %>%
  drop_na()
```

Appendix B - Prediction Models

This appendix provides the code used to develop both the regression and classification models.

Regression

```
# Baseline model
lr_baseline <- lm(median_outage_duration_hr ~ ., data = reg_train)

baseline_results <-
  eval_results(
    lr_baseline,
    reg_train$median_outage_duration_hr,
    lr_baseline$fitted.values
  )

baseline.cv <-
    cvFit(
    lr_baseline,
    data = reg_train,
    y = reg_train$median_outage_duration_hr,
    K = 10
  )
```

```
# Apply transformations to heavy tailed covariates
reg_train_transform <-
  reg_train %>%
   mutate(
    prop_latino = log10(prop_latino + 1),
    prop_less_than_hs = log10(prop_less_than_hs + 1)
  )
```

```
# Transformation model
lr_transform <- lm(median_outage_duration_hr ~ ., data = reg_train_transform)

transform_results <-
  eval_results(
    lr_transform,
    reg_train_transform$median_outage_duration_hr,
    lr_transform$fitted.values
  )

transform.cv <-
    cvFit(
    lr_transform,
    data = reg_train_transform,
    y = reg_train_transform$median_outage_duration_hr,
    K = 10
  )
```

```
# All interactions model
lr_interact <- lm(median_outage_duration_hr ~ . + .:., data = reg_train)
```

```
interact_results <-
  eval_results(
    lr_interact,
    reg_train$median_outage_duration_hr,
    lr_interact$fitted.values
  )

interact.cv <-
    cvFit(
    lr_interact,
    data = reg_train,
    y = reg_train$median_outage_duration_hr,
    K = 10
  )
```

```
# Transformed data with all interactions model
lr_transform_interact <- lm(median_outage_duration_hr ~ . + .:., data = reg_train_transform)

transform_interact_results <-
  eval_results(
    lr_transform_interact,
    reg_train_transform$median_outage_duration_hr,
    lr_transform_interact$fitted.values
  )

transform_interact.cv <-
    cvFit(
    lr_transform_interact,
    data = reg_train_transform,
    y = reg_train_transform$median_outage_duration_hr,
    K = 10
  )
```

```
# More specific selected covariates
# find variables most correlated to each other
# (select correlations w/ abs > .5)
covariate_corr <-
  reg_train %>%
  select(where(~sd(., na.rm = TRUE) > 0)) %>%
  correlate() %>%
  stretch() %>%
  arrange(r) %>%
  filter(abs(r) > .5) %>%
  mutate(lead_y = lead(y)) %>%
  filter(x != lead_y) %>%
  select(x, y)
```

```
##
## Correlation method: 'pearson'
## Missing treated using: 'pairwise.complete.obs'
```

```
# Reformat in formula form
interaction_vars <-
```

```
  covariate_corr %>%
  unite(col = "interact", sep = ":") %>%
  unlist() %>%
  paste(collapse = " + ")

# Pulled list of top covariates
lr_selected_interact_transform <-
  lm(
    as.formula(paste("median_outage_duration_hr ~ . +", interaction_vars)),
    data = reg_train_transform
  )

log_selected_interact_transform_results <-
  eval_results(
    lr_selected_interact_transform,
    reg_train_transform$median_outage_duration_hr,
    lr_selected_interact_transform$fitted.values
  )

log_selected_interact_transform.cv <-
    cvFit(
    lr_selected_interact_transform,
    data = reg_train_transform,
    y = reg_train_transform$median_outage_duration_hr,
    K = 10
  )
```

```
y <- reg_train_transform$median_outage_duration_hr
x <- reg_train_transform %>% select(-median_outage_duration_hr) %>% data.matrix()

# Ridge
cv_ridge <- cv.glmnet(x, y, alpha = 0, lambda = lambdas, standardize = TRUE)
ridge_model <- glmnet(x, y, alpha = 0, lambda = cv_ridge$lambda.min)

ridge_predict <- predict(ridge_model, x)

ridge_results <-
  eval_results(
    ridge_model,
    reg_train_transform$median_outage_duration_hr,
    ridge_predict
  )

ridge.cv <-
  cvFit(
    cv_ridge,
    x,
    y = reg_train_transform$median_outage_duration_hr,
    K = 10
  )

# Lasso
cv_lasso <- cv.glmnet(x, y, alpha = 1, lambda = lambdas, standardize = TRUE)
```

```r
lasso_model <- glmnet(x, y, alpha = 1, lambda = cv_lasso$lambda.min)

lasso_predict <- predict(lasso_model, x)

lasso_results <-
  eval_results(
    lasso_model,
    reg_train_transform$median_outage_duration_hr,
    lasso_predict
  )

lasso.cv <-
  cvFit(
    cv_lasso,
    x,
    y = reg_train_transform$median_outage_duration_hr,
    K = 10
  )
```

```r
df <-
  tribble(
    ~model, ~RMSE, ~Rsquare, ~CVerror,
    "baseline", baseline_results$RMSE, baseline_results$Rsquare, baseline.cv[["cv"]],
    "transform", transform_results$RMSE, transform_results$Rsquare, transform.cv[["cv"]],
    "all interactions", interact_results$RMSE, interact_results$Rsquare, interact.cv[["cv"]],
    "transform + interactions", transform_interact_results$RMSE, transform_interact_results$Rsquare, tra
    "selected transform + interactions", log_selected_interact_transform_results$RMSE, log_selected_int
    "ridge", ridge_results$RMSE, ridge_results$Rsquare, ridge.cv[["cv"]],
    "lasso", lasso_results$RMSE, lasso_results$Rsquare, lasso.cv[["cv"]]
  )

#write_rds(df, "~/GitHub/neighborhood-outages/cleaned_data/reg_stats.rds")
```

```r
# Look at results of the model
logit_mod <-
  glm(
    above_median_cust_affected ~ .,
    data = acs_outages_class,
    family = binomial
  )
summary(logit_mod)
```

Baseline model (logistic regression)

```
##
## Call:
## glm(formula = above_median_cust_affected ~ ., family = binomial,
##     data = acs_outages_class)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.4979  -0.7019  -0.5208  -0.3062   2.6394
##
## Coefficients: (5 not defined because of singularities)
##                                                 Estimate Std. Error z value
## (Intercept)                                   -2.553e+00  1.342e+00  -1.903
## prop_white                                     3.038e+00  6.892e-01   4.408
## prop_black_or_african_american                 2.662e+00  1.281e+00   2.079
## prop_american_indian_and_alaska_native         9.435e+00  4.340e+00   2.174
## prop_asian                                     2.537e+00  6.821e-01   3.719
## prop_native_hawaiian_and_other_pacific_islander  1.258e+00  6.707e+00   0.188
## prop_some_other_race                           3.717e+00  9.133e+00   0.407
## prop_multi_racial                             -6.932e+00  2.892e+00  -2.397
## prop_latino                                           NA         NA      NA
## pop_density_sq_km                              3.334e-05  2.255e-05   1.478
## prop_eli                                      -2.342e+00  2.723e+00  -0.860
## prop_hi                                       -7.690e-01  1.289e+00  -0.597
## prop_li                                       -1.453e+00  2.051e+00  -0.708
## prop_mi                                        2.409e-01  2.032e+00   0.119
## prop_vli                                              NA         NA      NA
## prop_college                                   4.837e-01  1.147e+00   0.422
## prop_high_school                               2.329e-01  1.210e+00   0.193
## prop_less_than_hs                                     NA         NA      NA
## prop_owner                                    -3.270e-01  3.976e-01  -0.822
## prop_renter                                           NA         NA      NA
## rental_vacancy_rate                            1.163e+00  1.093e+00   1.064
## owner_vacancy_rate                             2.839e+00  2.606e+00   1.089
## prop_rural                                     4.940e-01  2.266e-01   2.180
## prop_urban                                            NA         NA      NA
## n_outages_sq_km                               -3.509e-02  7.773e-03  -4.515
##                                                Pr(>|z|)
## (Intercept)                                      0.0571 .
## prop_white                                     1.04e-05 ***
## prop_black_or_african_american                   0.0377 *
```

```
## prop_american_indian_and_alaska_native            0.0297 *
## prop_asian                                        0.0002 ***
## prop_native_hawaiian_and_other_pacific_islander   0.8512
## prop_some_other_race                              0.6840
## prop_multi_racial                                 0.0165 *
## prop_latino                                           NA
## pop_density_sq_km                                 0.1393
## prop_eli                                          0.3898
## prop_hi                                           0.5507
## prop_li                                           0.4788
## prop_mi                                           0.9056
## prop_vli                                              NA
## prop_college                                      0.6732
## prop_high_school                                  0.8473
## prop_less_than_hs                                     NA
## prop_owner                                        0.4108
## prop_renter                                           NA
## rental_vacancy_rate                               0.2874
## owner_vacancy_rate                                0.2759
## prop_rural                                        0.0292 *
## prop_urban                                            NA
## n_outages_sq_km                                6.33e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2157.5  on 2162  degrees of freedom
## Residual deviance: 1956.2  on 2143  degrees of freedom
## AIC: 1996.2
##
## Number of Fisher Scoring iterations: 5
```

AIC is 1995.3.

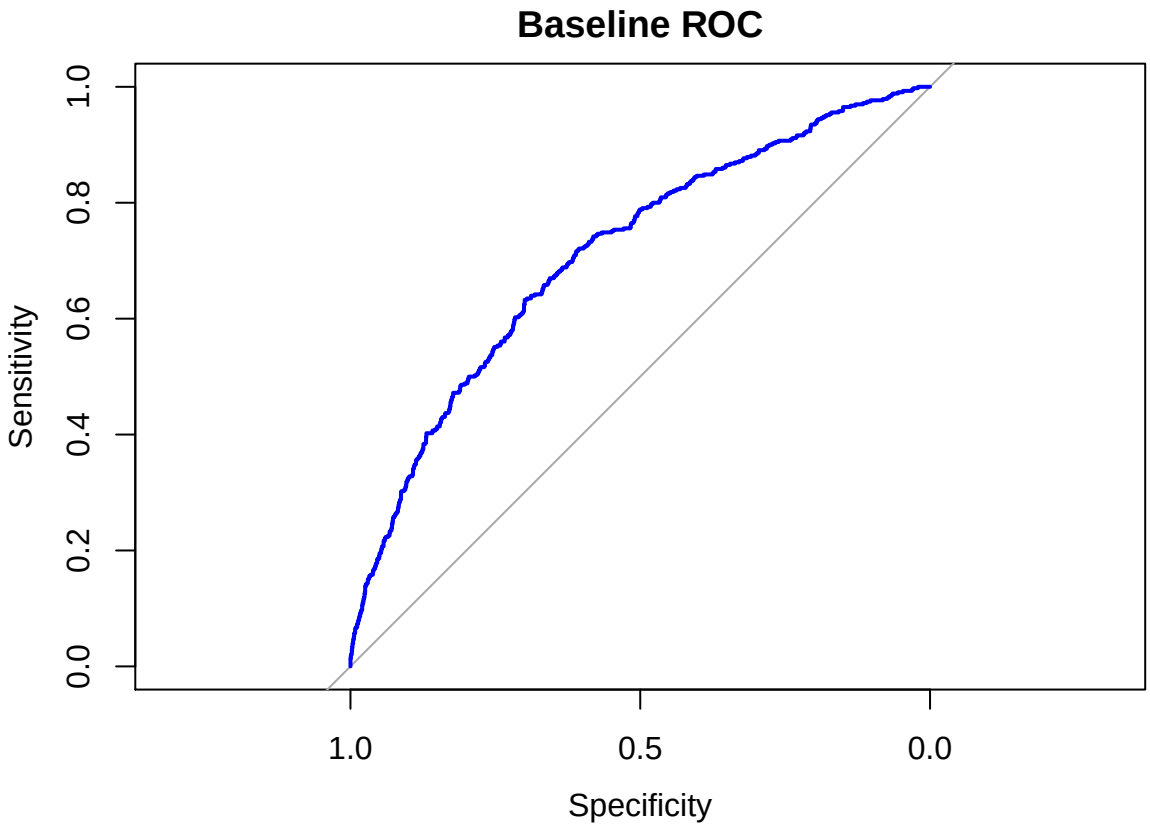
```
# Create confusion matrix
pred <-
  if_else(logit_mod$fitted.values > 0.5, "1", "0")

confusion_matrix <-
  table(acs_outages_class$above_median_cust_affected, pred)
rownames(confusion_matrix) <- c("Obs. 0", "Obs. 1")
colnames(confusion_matrix) <- c("Pred. 0", "Pred. 1")
confusion_matrix
```

```
##         pred
##          Pred. 0 Pred. 1
##   Obs. 0    1717      16
##   Obs. 1     401      29
```

```
# calculate confusion matrix scores
tn <- confusion_matrix[1,1]
fn <- confusion_matrix[2,1]
fp <- confusion_matrix[1,2]
```

```r
tp <- confusion_matrix[2,2]

base_01_loss <- (fp + fn) / nrow(acs_outages_class)
base_sensitivity <- tp / (fn + tp)
base_specificity <- tn / (tn + fp)
base_precision <- tp / (tp + fp)
base_typeI_error <- fp / (tn + fp)
base_typeII_error <- fn / (fn + tp)
base_false_discovery <- fp / (tp + fp)
```

```r
# Plot ROC Curve
roc(
  response = acs_outages_class$above_median_cust_affected,
  predictor = logit_mod$fitted.values,
  data = acs_outages_class,
  plot = TRUE,
  main = "Baseline ROC",
  col = "blue"
)
```




```
##
## Call:
## roc.default(response = acs_outages_class$above_median_cust_affected,     predictor = logit_mod$fitte
##
## Data: logit_mod$fitted.values in 1733 controls (acs_outages_class$above_median_cust_affected 0) < 43
## Area under the curve: 0.7106
```

```r
# determine AUC
auc(
  response = acs_outages_class$above_median_cust_affected,
  predictor = logit_mod$fitted.values,
  data = acs_outages_class
)
```

```
## Area under the curve: 0.7106
```

Major goal is optimize mean 0-1 loss and increase sensitivity. Logistic model is a good baseline, but I want to see if I can improve those metrics. Will use PCA to reduce covariates and eliminate some of the collinearity noticed in EDA. Then I'll take the factor loadings to conduct KNN classification. Will do a 10-fold cross-validation to ensure robustness of approach.

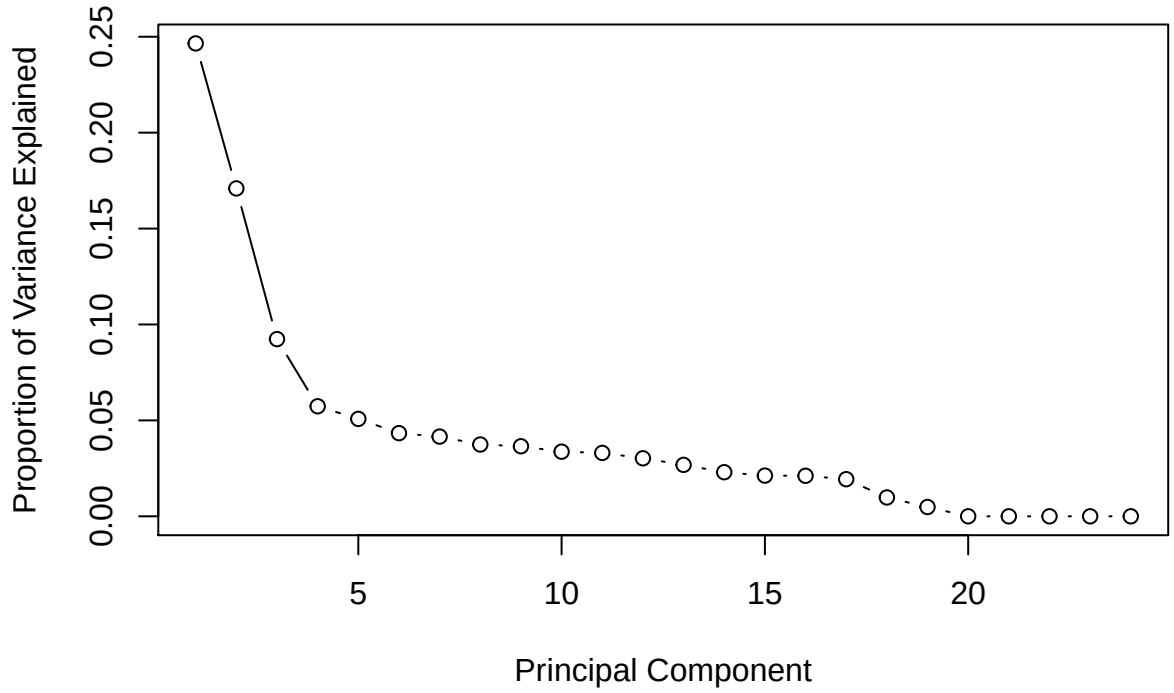
```r
# conduct PCA (to reduce data, especially collinearity)
# Find variance of each covariate
train_col_var <-
  acs_outages_class %>%
  select(-above_median_cust_affected) %>%
  mutate(across(everything(), stats::var)) %>%
  distinct() %>%
  pivot_longer(cols = everything(), names_to = "vars", values_to = "variance") %>%
  # filter out 0 variance covariates
  filter(variance != 0)

# remove zero variance covariates
class_filter <-
  acs_outages_class %>%
  # remove outcome var as well
  select(all_of(train_col_var$vars))

# Run PCA
pca <- prcomp(class_filter, scale = T, center = T)
summary(pca)
```

Final model (PCA-based K-nearest neighbors algorithm)

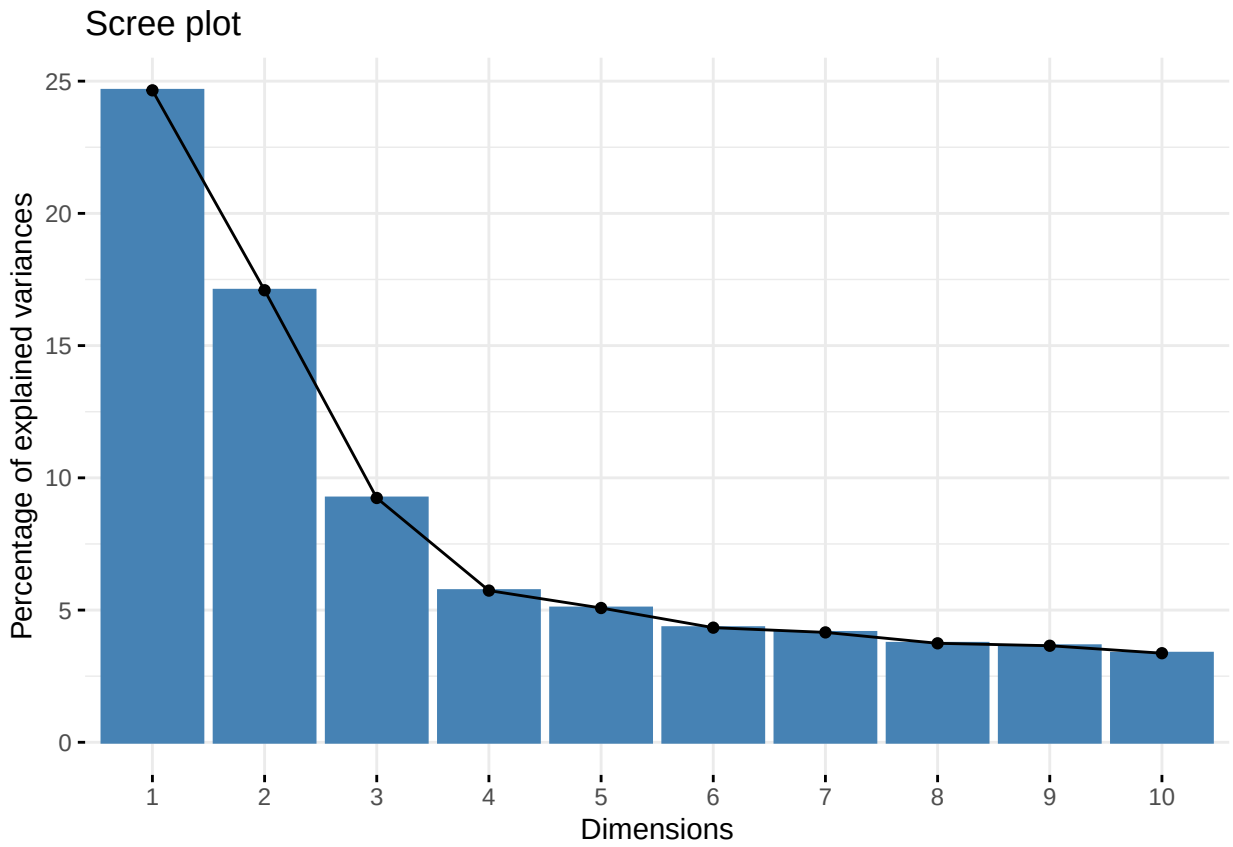
```
## Importance of components:
##                           PC1    PC2     PC3     PC4     PC5     PC6     PC7
## Standard deviation     2.4324 2.0254 1.48899 1.17335 1.10399 1.02003 0.99866
## Proportion of Variance 0.2465 0.1709 0.09238 0.05736 0.05078 0.04335 0.04155
## Cumulative Proportion  0.2465 0.4174 0.50983 0.56720 0.61798 0.66133 0.70289
##                           PC8     PC9    PC10    PC11    PC12    PC13    PC14
## Standard deviation     0.94780 0.93613 0.89918 0.89066 0.85196 0.80226 0.74272
## Proportion of Variance 0.03743 0.03651 0.03369 0.03305 0.03024 0.02682 0.02298
## Cumulative Proportion  0.74032 0.77683 0.81052 0.84357 0.87382 0.90064 0.92362
##                           PC15    PC16    PC17    PC18    PC19      PC20
## Standard deviation     0.71380 0.71199 0.68150 0.48556 0.34127 1.344e-15
## Proportion of Variance 0.02123 0.02112 0.01935 0.00982 0.00485 0.000e+00
## Cumulative Proportion  0.94485 0.96597 0.98532 0.99515 1.00000 1.000e+00
##                           PC21      PC22      PC23      PC24
## Standard deviation     7.241e-16 6.261e-16 5.094e-16 3.306e-16
```

```
## Proportion of Variance 0.000e+00 0.000e+00 0.000e+00 0.000e+00
## Cumulative Proportion  1.000e+00 1.000e+00 1.000e+00 1.000e+00
```

```r
# compute standard deviation of each principal component
std_dev <- pca$sdev
# compute variance
pr_var <- std_dev^2
# proportion of variance explained
prop_var <- pr_var/sum(pr_var)

# generate scree plot
plot(
  prop_var,
  xlab = "Principal Component",
  ylab = "Proportion of Variance Explained",
  type = "b"
)
```




```r
# Plot Eigenvalue variance from PCA
factoextra::fviz_eig(pca)
```



Around 7-8 components explains about 75% of variance.

```
# PCA results
pca_table <-
  data.frame(
    above_median_cust_affected = acs_outages_class[,1] %>% pull(), pca$x
  ) %>%
  #filter to the first 7 components
  select(above_median_cust_affected:PC7) %>%
  bind_cols()
```

```
set.seed(243)
# Do 10-fold CV KNN
# Set train control
knn_control <-
  trainControl(
    method = "cv",
    number = 10
  )

# Evaluate accuracy of KNN classifiers
knn_fit <-
  train(
    as.factor(above_median_cust_affected) ~ .,
    method = "knn",
    # Limit to k = sqrt(n)
    tuneGrid = expand.grid(k = 1:sqrt(nrow(pca_table))),
```

```
    trControl = knn_control,
    metric = "Accuracy",
    data = pca_table
  )
```

```
# create confusion matrix
confusion_matrix_knn <- confusionMatrix(knn_fit, positive = 1)
confusion_matrix_knn <- confusion_matrix_knn$table %>% t()
confusion_matrix_knn
```

```
##          Prediction
## Reference         0         1
##         0 79.010633  1.109570
##         1 17.984281  1.895515
```

```
# calculate confusion matrix scores
tn <- confusion_matrix_knn[1,1]
fn <- confusion_matrix_knn[2,1]
fp <- confusion_matrix_knn[1,2]
tp <- confusion_matrix_knn[2,2]

final_01_loss <- (fp + fn) / nrow(acs_outages_class)
final_sensitivity <- tp / (fn + tp)
final_specificity <- tn / (tn + fp)
final_precision <- tp / (tp + fp)
final_typeI_error <- fp / (tn + fp)
final_typeII_error <- fn / (fn + tp)
final_false_discovery <- fp / (tp + fp)
```

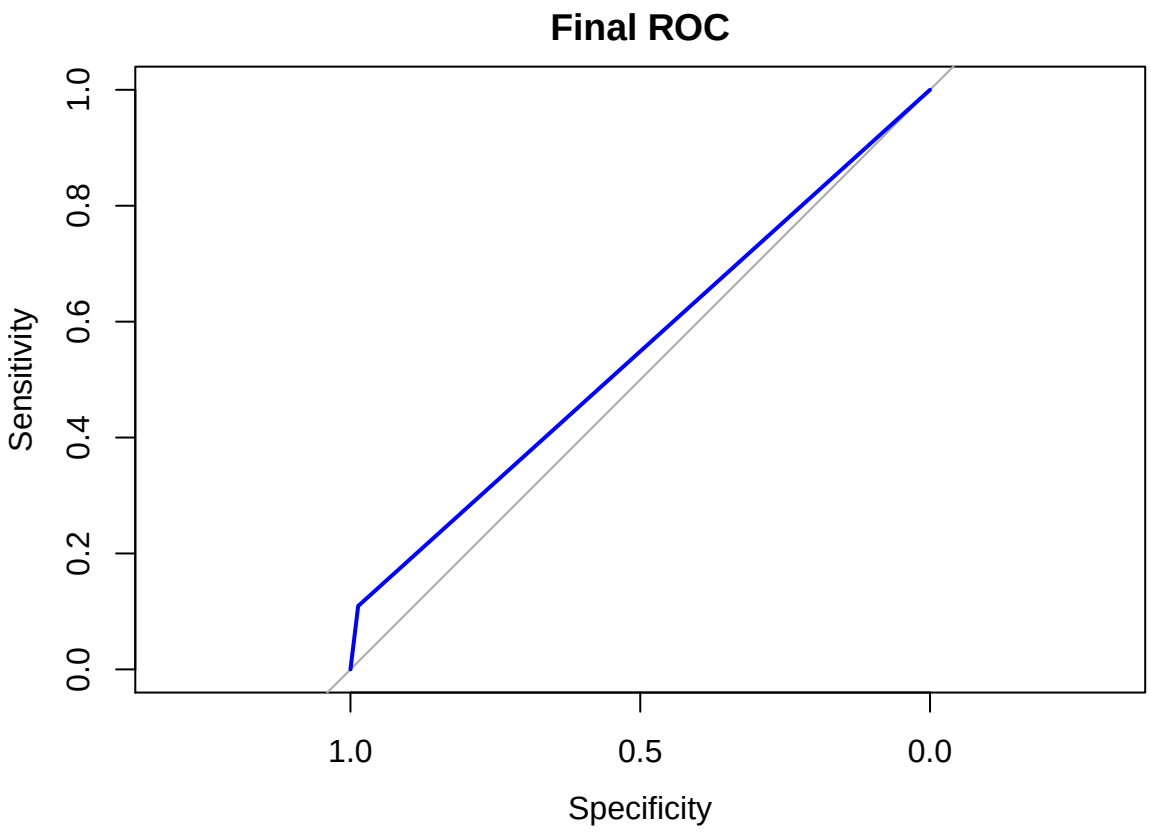
```
# confusion matrix comparison
confusion_matrix_comparison <-
  tibble(
    metric =
      c("Mean 0-1 Loss", "Precision", "Sensitivity", "Specificity",
        "Type I Error Rate", "Type II Error Rate", "False Discovery Rate"),
    base =
      c(base_01_loss, base_precision, base_sensitivity, base_specificity,
        base_typeI_error, base_typeII_error, base_false_discovery),
    final =
      c(final_01_loss, final_precision, final_sensitivity, final_specificity,
        final_typeI_error, final_typeII_error, final_false_discovery)
  ) %>%
  kable(caption = "Above Median Customer Affected Confusion Matrix Metrics") %>%
  kableExtra::kable_classic() %>%
  #format for markdown
  kable_styling(latex_options="scale_down")
confusion_matrix_comparison
```

Some metrics take a hit (like precision and specificity), but Sensitivity increased a good deal and the Mean 0-1 Loss improved as well - which were our choice outcomes. Want the model to detect as many true positives as possible.

Table 4: Above Median Customer Affected Confusion Matrix Metrics

metric	base	final
Mean 0-1 Loss	0.1927878	0.0088275
Precision	0.6444444	0.6307692
Sensitivity	0.0674419	0.0953488
Specificity	0.9907675	0.9861512
Type I Error Rate	0.0092325	0.0138488
Type II Error Rate	0.9325581	0.9046512
False Discovery Rate	0.3555556	0.3692308

```
# Plot ROC Curve
roc(
  response = pca_table$above_median_cust_affected,
  predictor = as.numeric(predict(knn_fit)),
  data = pca_table,
  plot = TRUE,
  main = "Final ROC",
  col = "blue"
)
```




```
##
## Call:
## roc.default(response = pca_table$above_median_cust_affected,     predictor = as.numeric(predict(knn_
##
## Data: as.numeric(predict(knn_fit)) in 1733 controls (pca_table$above_median_cust_affected 0) < 430 ca
## Area under the curve: 0.548
```

```
# determine AUC
auc(
  response = pca_table$above_median_cust_affected,
  predictor = as.numeric(predict(knn_fit)),
  data = pca_table
)
```

```
## Area under the curve: 0.548
```

Clearly AUC and ROC take a hit, but this was not the goal of the model. Wanted to reduce mean 0-1 loss and increase sensitivity above all. This model is likely biased, but the flag only appears 20% of the time I believe such bias is justified. This is especially true because I wanted to optimize the sensitivity and accuracy of the model.

Appendix C - Data Processing

This appendix provides the scripts used in processing, cleaning, and merging data.

ACS Clean Script

```
# Setup ------------------------------------------------------------------------
# Packages:
library(tidyverse)
library(sf)
library(tigris)
library(tidycensus)
options(tigris_use_cache = TRUE)

# Directories:
homedir <- "E:/neighborhood-outages/"
workdir <- "raw_data/"
savedir <- "cleaned_data/"
setwd(homedir)

# Import data:

# Parameters:

# Main Script ------------------------------------------------------------------

# Load available variables for 2018 ACS
acs_vars <- load_variables(year = 2018, dataset = "acs5", cache = TRUE)
dec_vars <- load_variables(year = 2010, dataset = "sf1", cache = TRUE)

# load land area for each census tract
ca_land_area <-
  tracts("CA") %>%
  st_drop_geometry() %>%
  select(GEOID, ALAND) %>%
  # convert from square meters to square kilometer
  transmute(
    GEOID = GEOID,
    land_area_sq_km = ALAND / 1000000
  )

# Read in ACS data
## ACS race data (2018)
acs_race <-
  get_acs(
    geography = "tract",
    table = "B03002",
    year = 2018,
    cache_table = TRUE,
    state = "CA"
  ) %>%
  left_join(acs_vars, by = c("variable" = "name")) %>%
  select(GEOID, label, estimate) %>%
  mutate(
```

```
    # convert names to better form
    label = str_to_lower(str_replace_all(label, "!!|\\s+", "_")),
    # short estimate to est
    label = str_replace_all(label, "estimate", "est"),
    # remove unnecessary character strings
    label = str_remove_all(label, "est_total_not_hispanic_or_latino_|_alone")
   ) %>%
  # remove total non-hispanic and hispanic by race
  filter(!str_detect(label, "est_total_not_hispanic_or_latino|or_latino_")) %>%
  # pivot table
  pivot_wider(
    names_from = label,
    values_from = estimate
  ) %>%
  # rename multiracial category
  rename(multi_racial = two_or_more_races) %>%
  # select out separated multiracial categories
  select(-starts_with("two_or_more")) %>%
  # create proportions by tract
  mutate(
    across(
      where(is.double),
      ~ . / est_total,
      .names = "prop_{.col}"
    )
  ) %>%
  # select for just tract ID, total population, and proportions
  select(GEOID, est_total, starts_with("prop")) %>%
  select(-prop_est_total) %>%
  rename(prop_latino = prop_est_total_hispanic_or_latino) %>%
  # join with land area to create density stat
  left_join(ca_land_area, by = "GEOID") %>%
  mutate(pop_density_sq_km = est_total / land_area_sq_km) %>%
  # drop total population variable
  select(-est_total)

## ACS income data (2018)
### Read in median income by tract
acs_median_income <-
  get_acs(
    geography = "county",
    variables = "B06011_001",
    year = 2018,
    cache_table = TRUE,
    state = "CA"
  ) %>%
  select(GEOID, ami = estimate)

## ACS income data (2018)
acs_income <-
  get_acs(
    geography = "tract",
    table = "B19001",
```

```
  year = 2018,
  cache_table = TRUE,
  state = "CA"
) %>%
left_join(acs_vars, by = c("variable" = "name")) %>%
select(GEOID, label, estimate) %>%
mutate(
  # convert names to better form
  label = str_to_lower(str_replace_all(label, "!!|\\s+", "_")),
  # short estimate to est
  label = str_replace_all(label, "estimate", "est"),
  est_total = if_else(label == "est_total", estimate, NA_real_)
) %>%
group_by(GEOID) %>%
fill(everything(), .direction = "down") %>%
ungroup() %>%
filter(label != "est_total") %>%
mutate(
  label = str_extract(label, "\\$\\d+,\\d+$"),
  label = str_remove_all(label, "\\$|[:punct:]"),
  label = if_else(is.na(label), 500000, as.double(label)),
  county = str_extract(GEOID, "^\\d{5}")
) %>%
rename(upper_income_limit = label) %>%
left_join(acs_median_income, by = c("county" = "GEOID")) %>%
# construct the income categories
mutate(
  income_category =
    case_when(
      upper_income_limit <= (.30 * ami)       ~ "eli",
      upper_income_limit <= (.50 * ami) &
        upper_income_limit > (.30 * ami)      ~ "vli",
      upper_income_limit <= (.80 * ami) &
        upper_income_limit > (.50 * ami)      ~ "li",
      upper_income_limit <= (1.20 * ami) &
        upper_income_limit > (.80 * ami)      ~ "mi",
      upper_income_limit > (1.20 * ami)       ~ "hi",
      TRUE                                    ~ NA_character_
    )
) %>%
select(GEOID, estimate, income_category, est_total) %>%
group_by(GEOID, income_category) %>%
mutate(estimate = sum(estimate, na.rm = TRUE)) %>%
slice(1) %>%
ungroup() %>%
# change shape of data
pivot_wider(names_from = income_category, values_from = estimate) %>%
# create proportions by tract
mutate(
  across(
    eli:vli,
    ~ . / est_total,
    .names = "prop_{.col}"
```

```
    )
  ) %>%
  # select for just tract ID, total population, and proportions
  select(GEOID, starts_with("prop")) %>%
  mutate(across(where(is.double), ~ replace_na(.x, replace = 0)))

rm(acs_median_income)

## ACS tenure data (2018)
acs_tenure <-
  get_acs(
    geography = "tract",
    table = "B25003",
    year = 2018,
    cache_table = TRUE,
    state = "CA"
  ) %>%
  left_join(acs_vars, by = c("variable" = "name")) %>%
  select(GEOID, label, estimate) %>%
  mutate(
    # convert names to better form
    label = str_to_lower(str_replace_all(label, "!!|\\s+", "_")),
    # short estimate to est
    label = str_replace_all(label, "estimate", "est"),
    est_total = if_else(label == "est_total", estimate, NA_real_)
  ) %>%
  group_by(GEOID) %>%
  fill(everything(), .direction = "down") %>%
  ungroup() %>%
  filter(label != "est_total") %>%
  mutate(label = str_remove_all(label, "est_total_|_occupied")) %>%
  pivot_wider(names_from = label, values_from = estimate) %>%
  # create proportions by tract
  mutate(
    across(
      owner:renter,
      ~ . / est_total,
      .names = "prop_{.col}"
    )
  )

## ACS college-educated data (2018)
acs_education <-
  get_acs(
    geography = "tract",
    table = "B15003",
    year = 2018,
    cache_table = TRUE,
    state = "CA"
  ) %>%
  left_join(acs_vars, by = c("variable" = "name")) %>%
  select(GEOID, label, estimate) %>%
  mutate(
```

```
    # convert names to better form
    label = str_to_lower(str_replace_all(label, "!!|\\s+", "_")),
    # short estimate to est
    label = str_replace_all(label, "estimate", "est"),
    est_total = if_else(label == "est_total", estimate, NA_real_)
  ) %>%
  group_by(GEOID) %>%
  fill(everything(), .direction = "down") %>%
  ungroup() %>%
  filter(label != "est_total") %>%
  mutate(
    label = str_remove_all(label, "est_total_"),
    label =
      case_when(
        str_detect(
          label, "associate|bachelor|master|professional|doctorate"
        )                                                    ~ "college",
        str_detect(
          label, "high_school|ged_or_|some_college"
        )                                                    ~ "high_school",
        TRUE                                                 ~ "less_than_hs"
      )
  ) %>%
  group_by(GEOID, label) %>%
  mutate(estimate = sum(estimate, na.rm = TRUE)) %>%
  slice(1) %>%
  ungroup() %>%
  pivot_wider(names_from = label, values_from = estimate) %>%
  # create proportions by tract
  mutate(
    across(
      college:less_than_hs,
      ~ . / est_total,
      .names = "prop_{.col}"
    )
  ) %>%
  # select for just tract ID, total population, and proportions
  select(GEOID, starts_with("prop"))

## ACS vacancy data (2018)
acs_vacancy <-
  get_acs(
    geography = "tract",
    table = "B25004",
    year = 2018,
    cache_table = TRUE,
    state = "CA"
  ) %>%
  left_join(acs_vars, by = c("variable" = "name")) %>%
  select(GEOID, label, estimate) %>%
  mutate(
    # convert names to better form
    label = str_to_lower(str_replace_all(label, "!!|\\s+", "_")),
```

```
    # short estimate to est
    label = str_replace_all(label, "estimate", "est")
  ) %>%
  filter(
    label %in%
      c(
        "est_total_for_rent", "est_total_rented,_not_occupied",
        "est_total_for_sale_only", "est_total_sold,_not_occupied"
      )
  ) %>%
  mutate(label = str_remove_all(label, "est_total_|,")) %>%
  pivot_wider(names_from = label, values_from = estimate) %>%
  left_join(acs_tenure, by = "GEOID") %>%
  transmute(
    GEOID = GEOID,
    rental_vacancy_rate = for_rent / (renter + rented_not_occupied + for_rent),
    owner_vacancy_rate = for_sale_only / (owner + sold_not_occupied + for_sale_only)
  )

### Update tenure
acs_tenure <-
  acs_tenure %>%
  # select for just tract ID, total population, and proportions
  select(GEOID, starts_with("prop"))

## Urban and rural population
dec_urban <-
  get_decennial(
    geography = "tract",
    variables = c("P002001", "P002002", "P002005"),
    year = 2010,
    cache_table = TRUE,
    state = "CA"
  ) %>%
  left_join(dec_vars, by = c("variable" = "name")) %>%
  select(GEOID, label, value) %>%
  mutate(
    # convert names to better form
    label = str_to_lower(str_replace_all(label, "!!|\\s+", "_"))
  ) %>%
  pivot_wider(names_from = label, values_from = value) %>%
  mutate(
    prop_urban = total_urban / total,
    prop_rural = total_rural / total
  ) %>%
  select(GEOID, prop_rural, prop_urban)

# Merge all ACS data
acs_merged <-
  acs_race %>%
  left_join(acs_income, by = "GEOID") %>%
  left_join(acs_education, by = "GEOID") %>%
  left_join(acs_tenure, by = "GEOID") %>%
```

```
  left_join(acs_vacancy, by = "GEOID") %>%
  left_join(dec_urban)

# Save Results ------------------------------------------------------------
write_csv(
  acs_merged,
  file = paste0(homedir, savedir, "acs_clean.csv")
)

rm(
  acs_education, acs_income, acs_merged, acs_race, acs_tenure,
  acs_vacancy, acs_vars, ca_land_area, dec_urban, dec_vars
)
```

Outage Clean Script

```
# Setup ------------------------------------------------------------------
# Packages:
library(tidyverse)
library(sf)
library(tigris)
library(lubridate)
options(tigris_use_cache = TRUE)

# Directories:
homedir <- "E:/neighborhood-outages/"
workdir <- "raw_data/"
savedir <- "cleaned_data/"
setwd(homedir)

# Import data:
outages <- read_csv(paste0(homedir, workdir, "outages_expanded.csv"))

# Parameters:

# Main Script ------------------------------------------------------------

# Read in California block groups
ca_tracts <- tracts(state = "CA")

# Filter outages and geocode lat/long to Census tract
outages_filter <-
  outages %>%
  # Extract possible duration hours, min/max est affected, lat/long
  select(
    possible_duration_hours,
    min_estCustAffected,
    max_estCustAffected,
    latitude,
    longitude
  ) %>%
  # determine mean customers affected
  mutate(
    mean_cust_affected = (min_estCustAffected + max_estCustAffected) / 2
  ) %>%
  # convert to sf object
  st_as_sf(
    coords = c("longitude", "latitude"), crs = st_crs(ca_tracts)
  ) %>%
  # join to CA block groups
  st_join(ca_tracts) %>%
  # drop geometry
  st_drop_geometry() %>%
  # select out unnecessary columns
  select(
    GEOID, outage_duration_hr = possible_duration_hours, mean_cust_affected
  )
```

```r
# Remove unnecessary objects
rm(ca_tracts, outages)

# Save Results ------------------------------------------------------------
write_csv(
  outages_filter,
  file = paste0(homedir, savedir, "outages_clean.csv")
)
```

ACS and Outage Merge Script

```
# Setup ------------------------------------------------------------------
# Packages:
library(tidyverse)
#library(here)

# Directories:
homedir <- "E:/neighborhood-outages/"
workdir <- "cleaned_data/"
savedir <- "cleaned_data/"
setwd(homedir)

# Parameters
outages_filepath <- paste0(homedir, workdir, "outages_clean.csv")
acs_filepath <- paste0(homedir, workdir, "acs_clean.csv")

#outages_filepath <- here("cleaned_data/outages_clean.csv")
#acs_filepath <- here("cleaned_data/acs_clean.csv")

# Import data:
outages <- read_csv(outages_filepath)
acs <- read_csv(acs_filepath)

# Parameters:
set.seed(572)

# Main Script ------------------------------------------------------------

outages_grouped <-
  outages %>%
  group_by(GEOID) %>%
  summarise(
    median_outage_duration_hr = median(outage_duration_hr),
    median_mean_cust_affected = median(mean_cust_affected),
    num_outages = n()
  ) %>%
  ungroup() %>%
  mutate(
    above_median_cust_affected =
      if_else(
        median_mean_cust_affected > median(median_mean_cust_affected), 1, 0
      )
  ) %>%
  # remove median of mean customers affected column
  select(-median_mean_cust_affected)

acs_outages <-
  outages_grouped %>%
  left_join(acs, by = "GEOID") %>%
  mutate(
    median_outage_duration_hr = replace_na(median_outage_duration_hr, 0),
    above_median_cust_affected = replace_na(above_median_cust_affected, 0),
```

```
    num_outages = replace_na(num_outages, 0),
    n_outages_sq_km = num_outages / land_area_sq_km
  ) %>%
  # drop unneeded variables
  select(-c(num_outages, land_area_sq_km)) %>%
  mutate(rowid = row_number())

# CA census tracts that we don't have outage data for, as they are likely
# not serviced by PG&E.
non_outage_tracts <-
  setdiff(acs$GEOID, outages_grouped$GEOID) %>%
  as_tibble() %>%
  rename(non_pge_tract = value)

# Create test and train sets

acs_outages_test <-
  acs_outages %>%
  slice_sample(prop = .2)

acs_outages_train <-
  acs_outages %>%
  filter(!(rowid %in% acs_outages_test$rowid)) %>%
  select(-rowid)

acs_outages_test <- acs_outages_test %>% select(-rowid)

rm(acs, outages, outages_grouped, acs_outages)

# Save Results ------------------------------------------------------------
## write test ACS outages data
write_csv(
  acs_outages_test,
  file = paste0(homedir, savedir, "acs_outages_test.csv")
)

## write train ACS outages data
write_csv(
  acs_outages_train,
  file = paste0(homedir, savedir, "acs_outages_train.csv")
)

## write non-outage tracts
write_csv(
  non_outage_tracts,
  file = paste0(homedir, savedir, "non_outage_tracts.csv")
)

rm(non_outage_tracts, acs_outages_test, acs_outages_train)
```

Appendix D - EDA

This appendix provides all the EDA conducted prior to model building.

Missing values

```
# NA values?
# Filter all the columns to exclude NA
no_na_df <-
  acs_outages %>%
  filter(across(everything(), ~ !is.na(.)))

acs_outages %>%
  filter(!(GEOID %in% no_na_df$GEOID)) %>%
  arrange(desc(prop_white))
```

```
## # A tibble: 15 x 27
##    GEOID median_outage_d~ above_median_cu~ prop_white prop_black_or_a~
##    <chr>            <dbl>            <dbl>      <dbl>            <dbl>
##  1 0600~             0.67                1      0.883           0
##  2 0607~             0.83                0      0.833           0
##  3 0608~             1.66                0      0.641           0.172
##  4 0605~             3                   1      0.629           0.0443
##  5 0605~             2.5                 0      0.548           0.220
##  6 0609~             3                   0      0.548           0.186
##  7 0611~            36.7                 0      0.539           0.0847
##  8 0603~             3.24                1      0.485           0.173
##  9 0605~             9.76                0      0.428           0.0544
## 10 0607~             1.33                0      0.403           0.339
## 11 0607~             1.33                0      0.402           0.0565
## 12 0607~             2.17                0      0.299           0.0787
## 13 0601~             2.5                 0      0.217           0.117
## 14 0607~             4.67                1      0.194           0.509
## 15 0609~             9.75                0     NA              NA
## # ... with 22 more variables: prop_american_indian_and_alaska_native <dbl>,
## #   prop_asian <dbl>, prop_native_hawaiian_and_other_pacific_islander <dbl>,
## #   prop_some_other_race <dbl>, prop_multi_racial <dbl>, prop_latino <dbl>,
## #   pop_density_sq_km <dbl>, prop_eli <dbl>, prop_hi <dbl>, prop_li <dbl>,
## #   prop_mi <dbl>, prop_vli <dbl>, prop_college <dbl>, prop_high_school <dbl>,
## #   prop_less_than_hs <dbl>, prop_owner <dbl>, prop_renter <dbl>,
## #   rental_vacancy_rate <dbl>, owner_vacancy_rate <dbl>, prop_rural <dbl>,
## #   prop_urban <dbl>, n_outages_sq_km <dbl>
```

Not that many NAs, should consider dropping.

```
# NULL values?
# Filter all the columns to exclude NA
no_null_df <-
  acs_outages %>%
  filter(across(everything(), ~ !is.null(.)))

acs_outages %>%
  filter(!(GEOID %in% no_null_df$GEOID)) %>%
  arrange(desc(prop_white))
```

```
## # A tibble: 0 x 27
## # ... with 27 variables: GEOID <chr>, median_outage_duration_hr <dbl>,
## #   above_median_cust_affected <dbl>, prop_white <dbl>,
## #   prop_black_or_african_american <dbl>,
## #   prop_american_indian_and_alaska_native <dbl>, prop_asian <dbl>,
## #   prop_native_hawaiian_and_other_pacific_islander <dbl>,
## #   prop_some_other_race <dbl>, prop_multi_racial <dbl>, prop_latino <dbl>,
## #   pop_density_sq_km <dbl>, prop_eli <dbl>, prop_hi <dbl>, prop_li <dbl>,
## #   prop_mi <dbl>, prop_vli <dbl>, prop_college <dbl>, prop_high_school <dbl>,
## #   prop_less_than_hs <dbl>, prop_owner <dbl>, prop_renter <dbl>,
## #   rental_vacancy_rate <dbl>, owner_vacancy_rate <dbl>, prop_rural <dbl>,
## #   prop_urban <dbl>, n_outages_sq_km <dbl>
```

No NULL values.

Correlations

```
### Generate correlation matrices with p-values ###

# remove continuous/outcome vars if sd < 0
corr_matrix <-
  acs_outages %>%
  select(all_of(continuous_vars), all_of(outcome_vars)) %>%
  select(where(~sd(., na.rm = TRUE) > 0))

# create correlation matrix
corr_matrix <- rcorr(as.matrix(corr_matrix))
```

```
## Warning in sqrt(1 - h * h): NaNs produced
```

```
corr_matrix <-
  # merge p-values
  bind_cols(
    as.data.frame(corr_matrix$r),
    as.data.frame(corr_matrix$P) %>% rename_all(~paste0(.x, "_p-value"))
  ) %>%
  # filter to relevant predictor and outcome variables
  select(all_of(outcome_vars), all_of(paste0(outcome_vars, "_p-value"))) %>%
  rownames_to_column(var = "predictor") %>%
  filter(
    predictor %in% c(continuous_vars, paste0(continuous_vars, "_p-value"))
  ) %>%
  column_to_rownames(var = "predictor") %>%
  select(sort(colnames(.))) %>%
  as.data.frame()
```

```
### Filter to highest correlated & statistical significant ###
# For each outcome variable, select significant correlations (in abs value)

# create list of variables to use as a filter
corr_top <- foreach(out = outcome_vars) %do% {
  out_p <- paste0(out, "_p-value")
  corr_matrix %>%
```

Table 5: Median Outage Duration Correlations

predictor	median_outage_duration_hr	median_outage_duration_hr_p-value
n_outages_sq_km	-0.1123572	0.0000001
prop_white	0.1120240	0.0000002
prop_american_indian_and_alaska_native	0.1059761	0.0000007
pop_density_sq_km	-0.0886237	0.0000345
prop_latino	-0.0880026	0.0000393
prop_less_than_hs	-0.0833460	0.0000989
prop_urban	-0.0664365	0.0019211
prop_rural	0.0664365	0.0019211
prop_black_or_african_american	-0.0621140	0.0037403
prop_owner	0.0579319	0.0068824
prop_renter	-0.0579319	0.0068824
prop_eli	-0.0551512	0.0100433
owner_vacancy_rate	0.0466866	0.0299138
prop_college	0.0453477	0.0343685

```
    select(!!out, !!out_p) %>%
    filter(!!sym(out_p) < 0.05) %>%
    arrange(desc(abs(.))) %>%
    # slice to the top 10 results
    # head(10) %>%
    # extract the rownames for filtering
    rename_all(~str_remove(., "outcome_")) %>%
    rownames_to_column(var = "predictor") %>%
    mutate(predictor = str_remove(predictor, "continuous_"))
}
names(corr_top) <- str_remove(outcome_vars, "outcome_")

corr_top$median_outage_duration_hr %>%
  kable(caption = "Median Outage Duration Correlations") %>%
  kableExtra::kable_classic() %>%
  #format for markdown
  kable_styling(latex_options="scale_down")
```

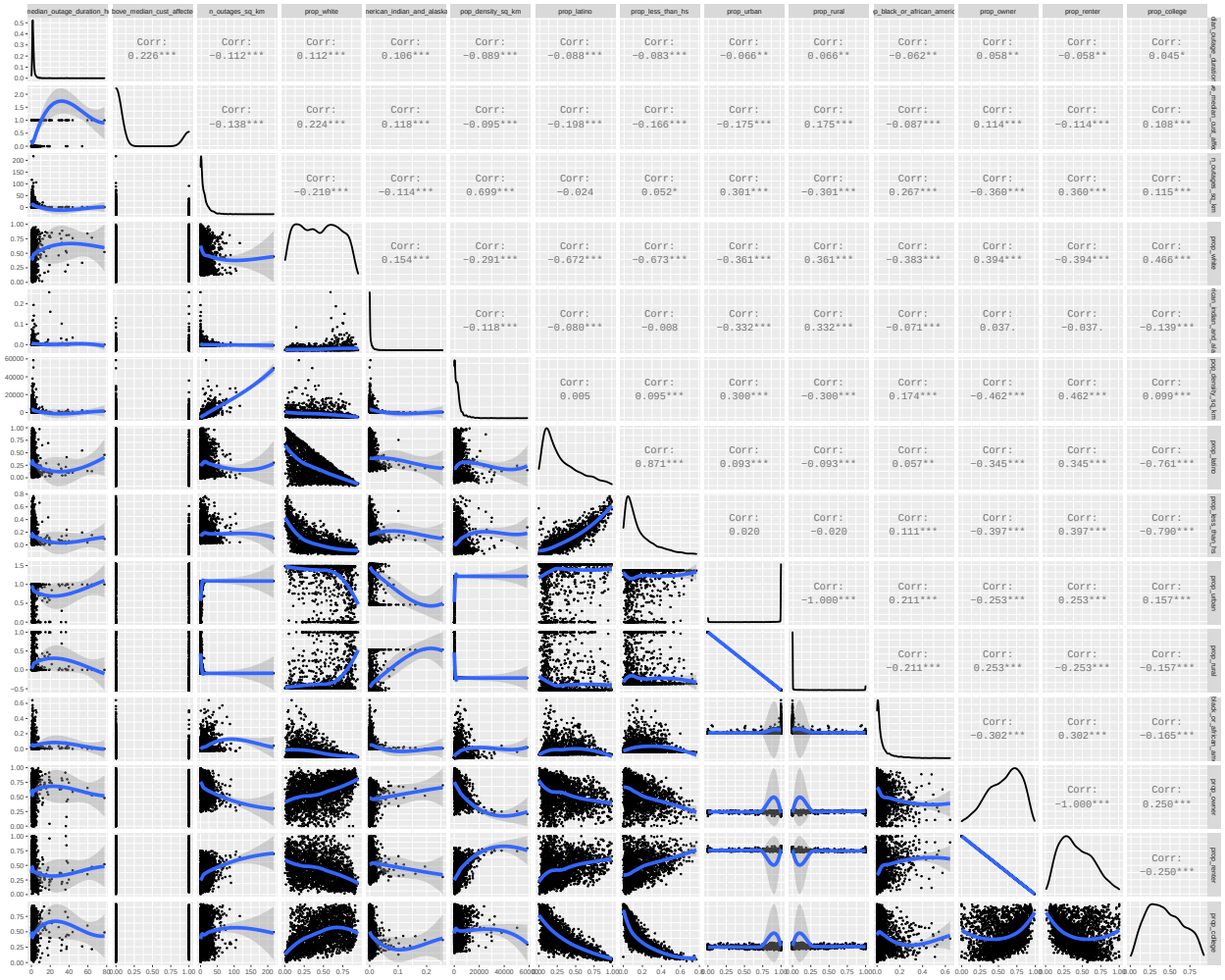
```
corr_top$above_median_cust_affected %>%
  kable(caption = "Above Median Customer Affected Correlations") %>%
  kableExtra::kable_classic() %>%
  #format for markdown
  kable_styling(latex_options="scale_down")
```

```
# create ggpairs scatterplot - looking at top 10 correlations
# visualize data to choose transformation


# make a function to plot generic data with points and a loess line
my_fn <- function(data, mapping, method="loess", ...){
  p <- ggplot(data = data, mapping = mapping) +
    geom_point(size = 0.01) +
    geom_smooth(method=method, formula = y ~ x, ...)
  return(p)
```

Table 6: Above Median Customer Affected Correlations

predictor	above_median_cust_affected	above_median_cust_affected_p-value
prop_white	0.2244869	0.0000000
prop_latino	-0.1981403	0.0000000
prop_urban	-0.1745335	0.0000000
prop_rural	0.1745335	0.0000000
prop_less_than_hs	-0.1662176	0.0000000
n_outages_sq_km	-0.1380306	0.0000000
prop_american_indian_and_alaska_native	0.1175222	0.0000000
prop_owner	0.1136678	0.0000001
prop_renter	-0.1136678	0.0000001
prop_college	0.1084843	0.0000004
pop_density_sq_km	-0.0946454	0.0000097
prop_black_or_african_american	-0.0870522	0.0000476
rental_vacancy_rate	0.0813124	0.0001469
prop_hi	0.0651032	0.0023675
prop_li	-0.0644803	0.0026070
owner_vacancy_rate	0.0548375	0.0107467
prop_eli	-0.0466111	0.0296128
prop_mi	-0.0446051	0.0373865

```
}

acs_outages %>%
  select(-GEOID) %>%
  # filter to significantly correlated variables
  select(
    all_of(outcome_vars),
    corr_top$median_outage_duration_hr$predictor %>% head(10),
    corr_top$above_median_cust_affected$predictor %>% head(10)
  ) %>%
  ggpairs(
    upper = list(continuous = wrap("cor", size = 2), size = 0.01),
    lower = list(continuous = my_fn, combo = wrap("facethist", binwidth = .1)),
    progress = FALSE
  ) +
  theme_grey(base_size = 5)
```



```
# find variables most correlated to each other
# (select correlations w/ abs > .5)
covariate_corr <-
  acs_outages %>%
  select(all_of(continuous_vars), all_of(outcome_vars)) %>%
  # drop prop owner
  select(-prop_owner) %>%
  select(where(~sd(., na.rm = TRUE) > 0)) %>%
  correlate() %>%
  stretch() %>%
  arrange(r) %>%
  filter(abs(r) > .5)
```

```
##
## Correlation method: 'pearson'
## Missing treated using: 'pairwise.complete.obs'
```

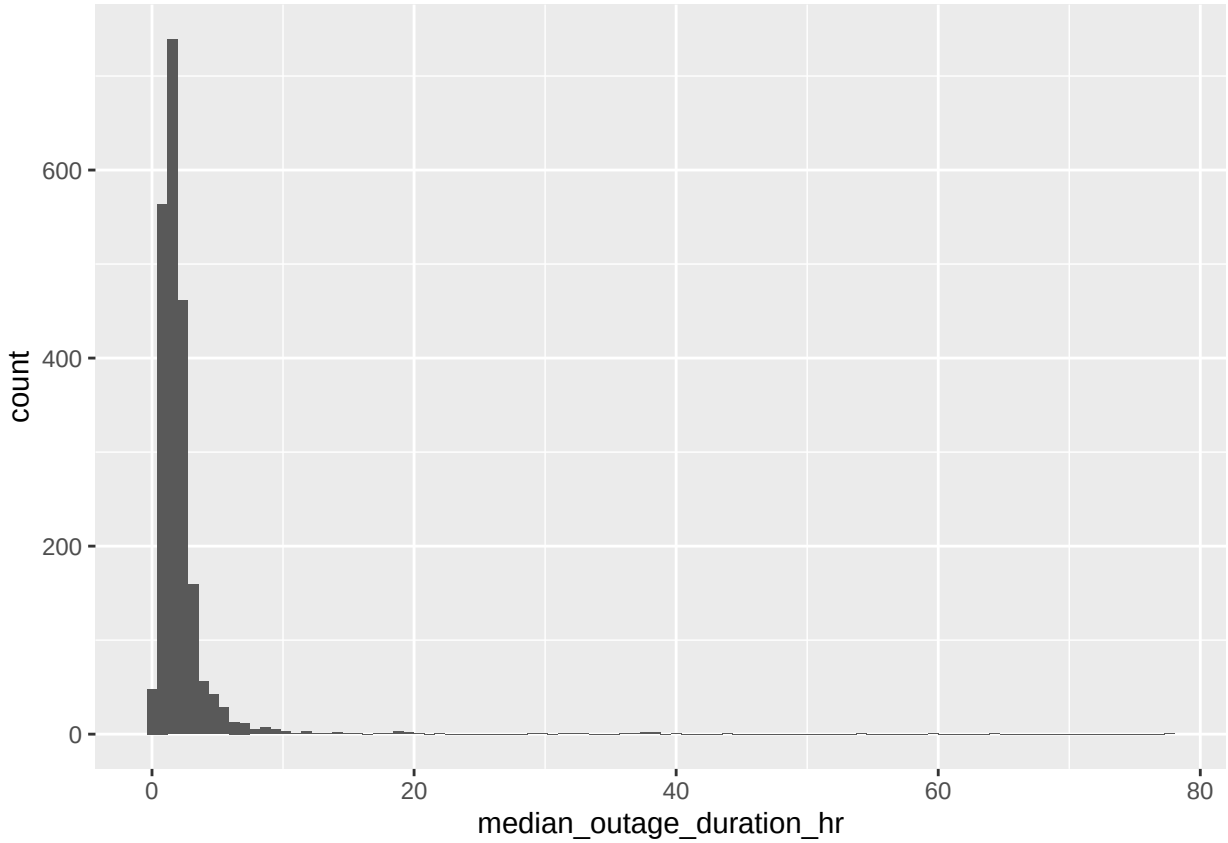
```
# reformat in formula form
interaction_vars <-
  covariate_corr %>%
  mutate(lead_y = lead(y)) %>%
  filter(x != lead_y) %>%
```

```
  select(x, y) %>%
  unite(col = "interact", sep = ":") %>%
  unlist() %>%
  paste(collapse = " + ")
covariate_corr
```

```
## # A tibble: 30 x 3
##    x                y                     r
##    <chr>            <chr>             <dbl>
##  1 prop_rural       prop_urban       -1
##  2 prop_urban       prop_rural       -1
##  3 prop_college     prop_less_than_hs -0.790
##  4 prop_less_than_hs prop_college     -0.790
##  5 prop_college     prop_high_school  -0.785
##  6 prop_high_school prop_college      -0.785
##  7 prop_latino      prop_college      -0.761
##  8 prop_college     prop_latino       -0.761
##  9 prop_hi          prop_vli          -0.741
## 10 prop_vli         prop_hi           -0.741
## # ... with 20 more rows
```

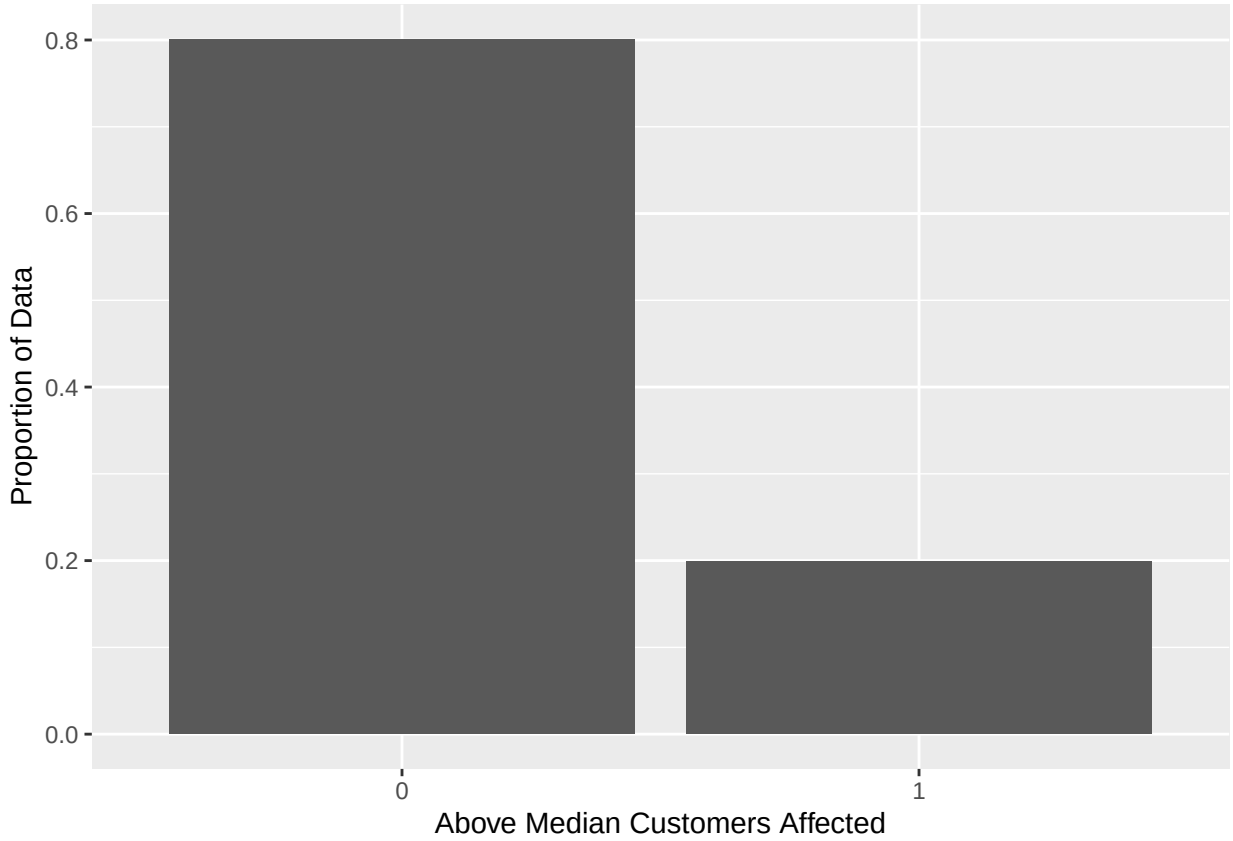
Median outage histogram

```
acs_outages %>%
  ggplot(aes(x = median_outage_duration_hr)) +
  geom_histogram(bins = 100)
```



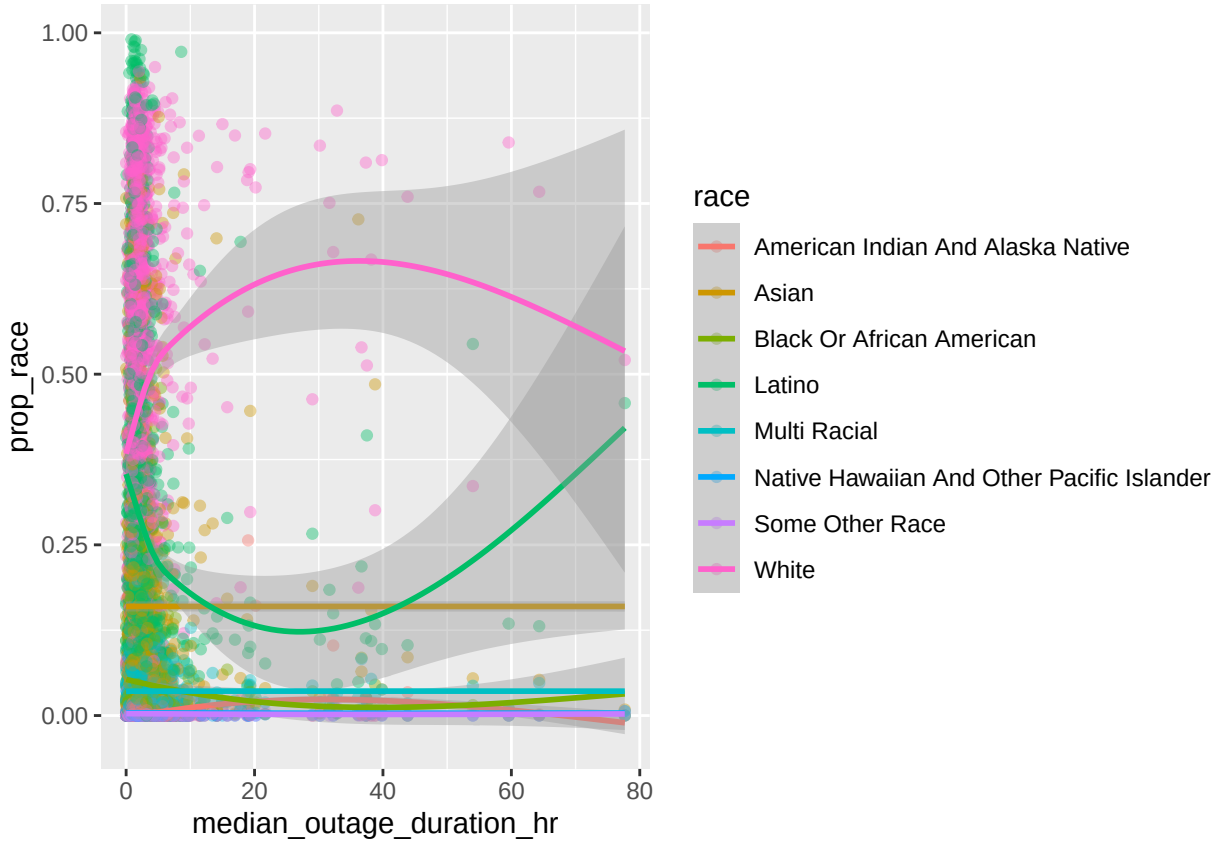
Above average customer affected distribution plot

```
acs_outages %>%
  ggplot(aes(x = as.factor(above_median_cust_affected))) +
  geom_bar(aes(y = (..count..)/sum(..count..))) +
  labs(
    x = "Above Median Customers Affected",
    y = "Proportion of Data"
  )
```



Bar plots

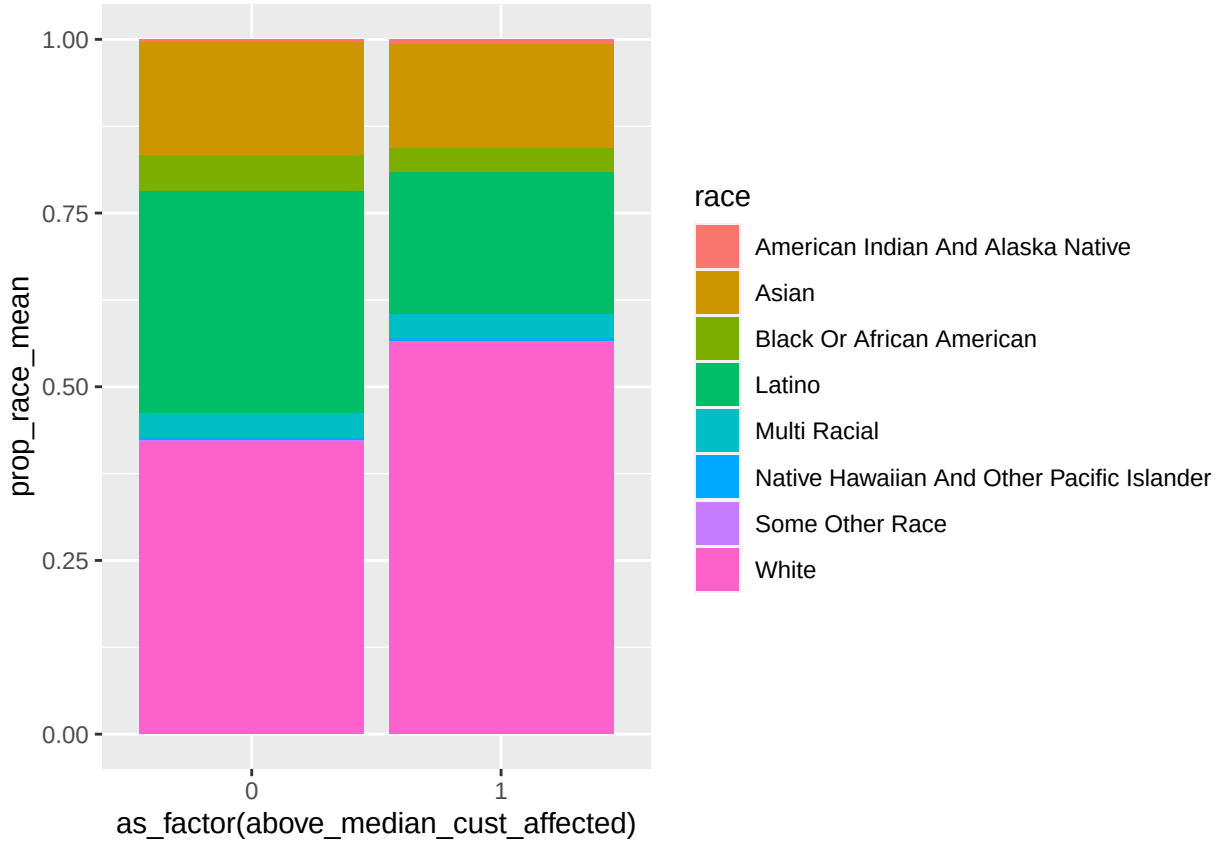
```
acs_outages %>%
  pivot_longer(
    cols = prop_white:prop_latino,
    names_to = "race",
    values_to = "prop_race"
  ) %>%
  mutate(
    race =str_to_title(str_replace_all(str_remove(race, "^prop_"), "_", " "))
  ) %>%
  select(all_of(outcome_vars), race, prop_race) %>%
  ggplot(aes(x = median_outage_duration_hr, y = prop_race, color = race)) +
  geom_point(alpha = .4) +
  geom_smooth()
```



Race

```
acs_outages %>%
  pivot_longer(
    cols = prop_white:prop_latino,
    names_to = "race",
    values_to = "prop_race"
  ) %>%
  mutate(
    race =str_to_title(str_replace_all(str_remove(race, "^prop_"), "_", " "))
  ) %>%
  select(all_of(outcome_vars), race, prop_race) %>%
  group_by(above_median_cust_affected, race) %>%
  summarise(prop_race_mean = mean(prop_race, na.rm = TRUE)) %>%
  ggplot(
    aes(
      x = as_factor(above_median_cust_affected),
      y = prop_race_mean,
      fill = race
    )
  ) +
  geom_col()
```

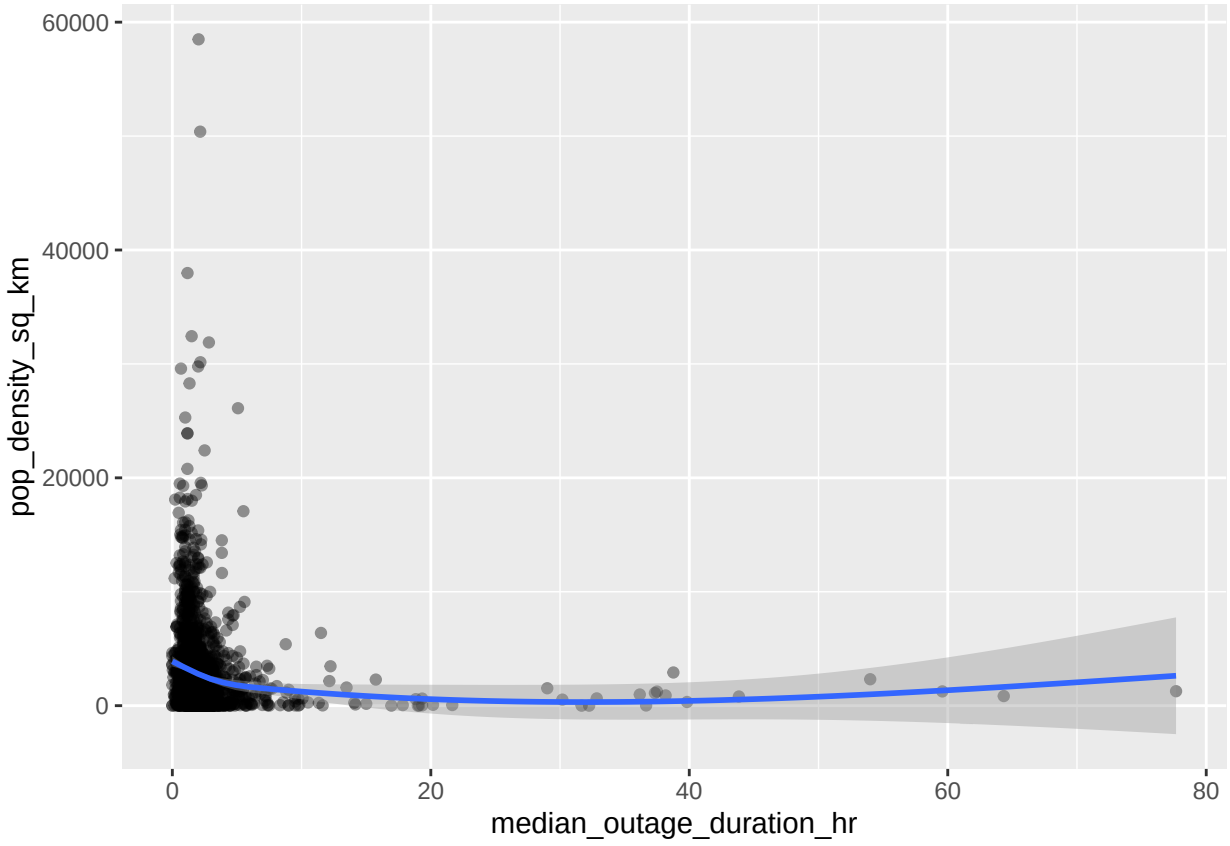
```
## `summarise()` regrouping output by 'above_median_cust_affected' (override with `.groups` argument)
```



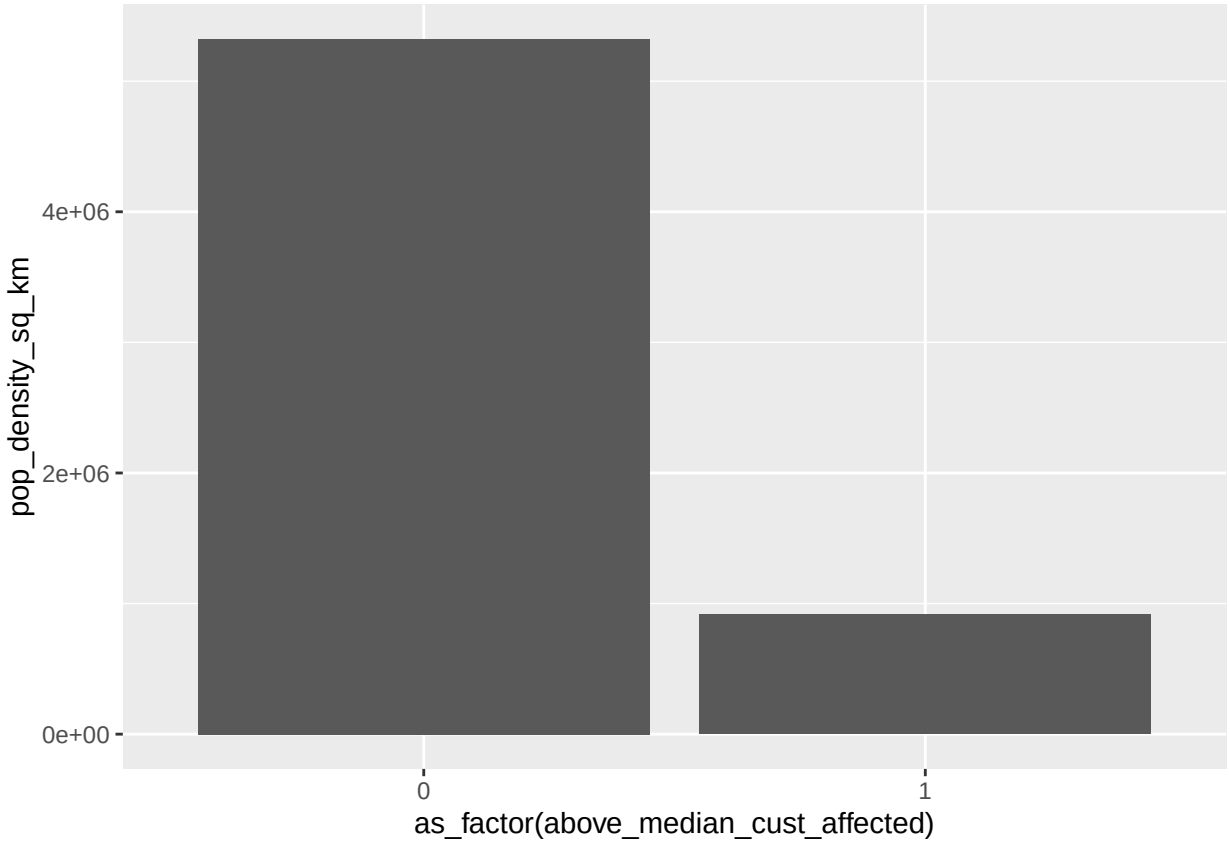

```
acs_outages %>%
  ggplot(aes(x = median_outage_duration_hr, y = pop_density_sq_km)) +
  geom_point(alpha = .4) +
  geom_smooth()
```

Population density

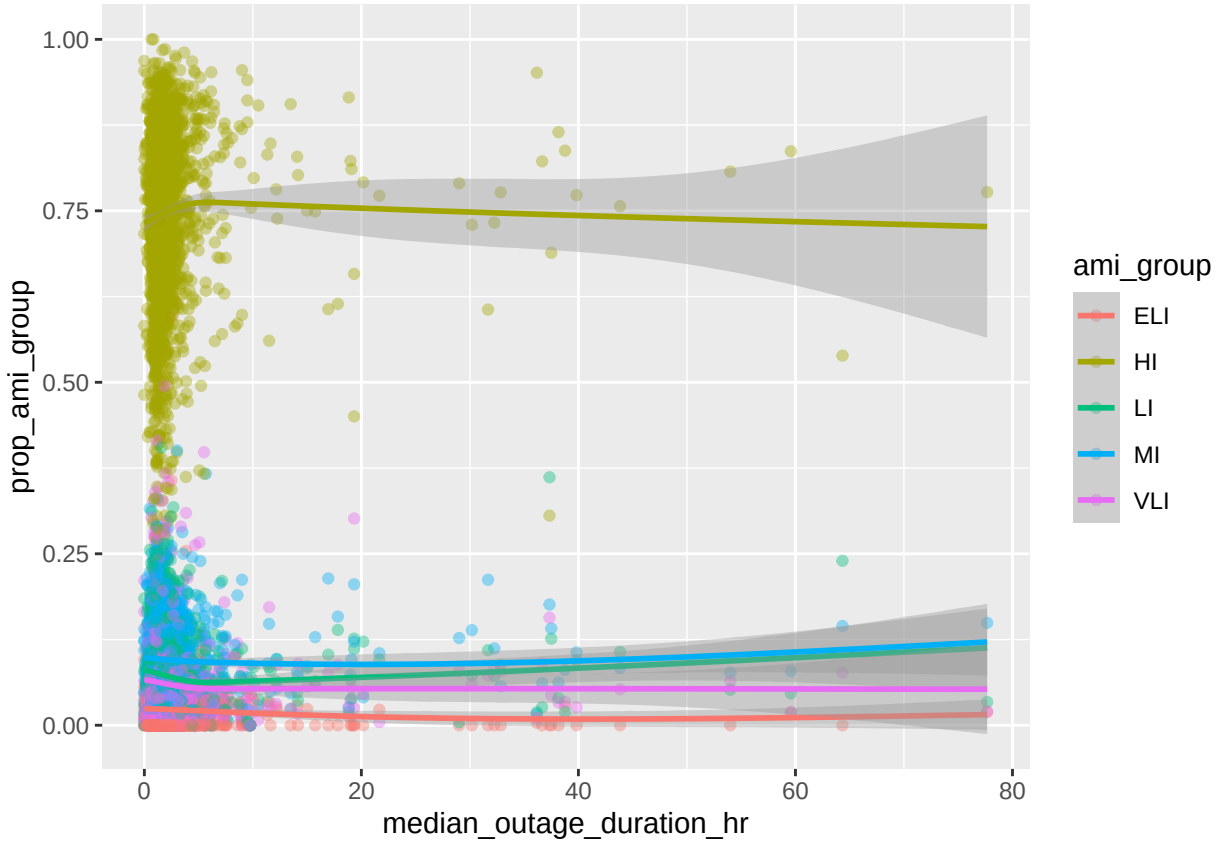
```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```




```
acs_outages %>%
  ggplot(
    aes(
      x = as_factor(above_median_cust_affected),
      y = pop_density_sq_km
    )
  ) +
  geom_col()
```

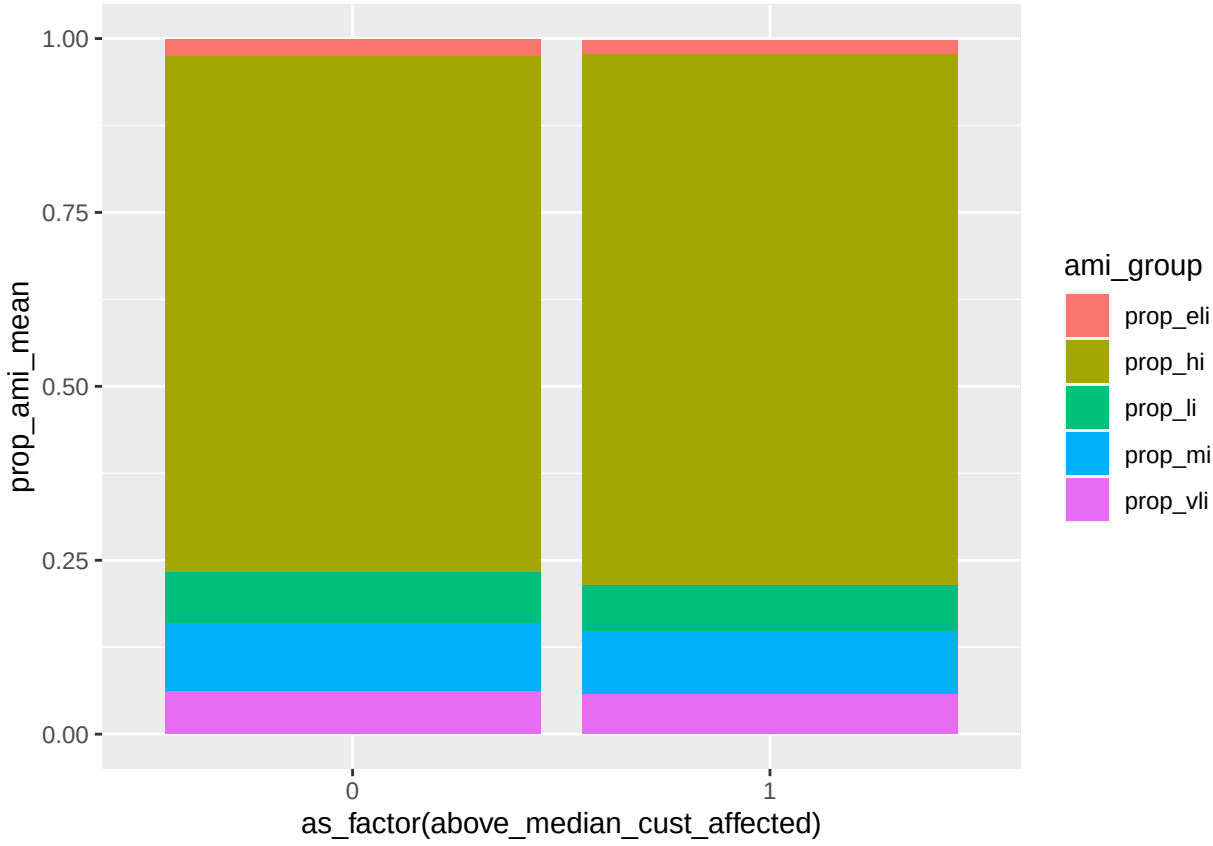



```
acs_outages %>%
  pivot_longer(
    cols = prop_eli:prop_vli,
    names_to = "ami_group",
    values_to = "prop_ami_group"
  ) %>%
  mutate(
    ami_group =
      str_to_upper(str_replace_all(str_remove(ami_group, "^prop_"), "_", " "))
  ) %>%
  select(all_of(outcome_vars), ami_group, prop_ami_group) %>%
  ggplot(
    aes(x = median_outage_duration_hr, y = prop_ami_group, color = ami_group)
  ) +
  geom_point(alpha = .4) +
  geom_smooth()
```

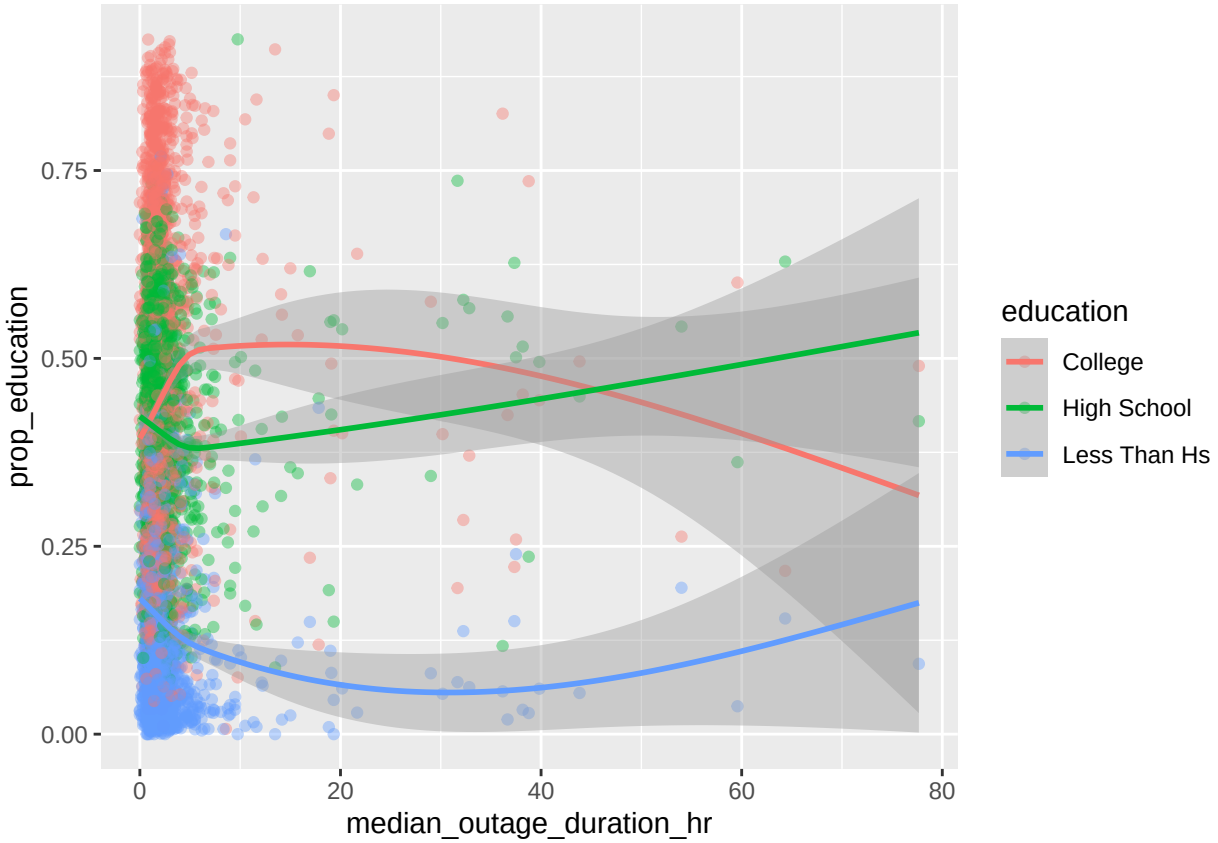


AMI Groups

```
acs_outages %>%
 pivot_longer(
    cols = prop_eli:prop_vli,
    names_to = "ami_group",
    values_to = "prop_ami_group"
 ) %>%
 mutate(
   race =
     str_to_upper(str_replace_all(str_remove(ami_group, "^prop_"), "_", " "))
 ) %>%
 select(all_of(outcome_vars), ami_group, prop_ami_group) %>%
 group_by(above_median_cust_affected, ami_group) %>%
 summarise(prop_ami_mean = mean(prop_ami_group, na.rm = TRUE)) %>%
 ggplot(
   aes(
     x = as_factor(above_median_cust_affected),
     y = prop_ami_mean,
     fill = ami_group
   )
 ) +
 geom_col()
```

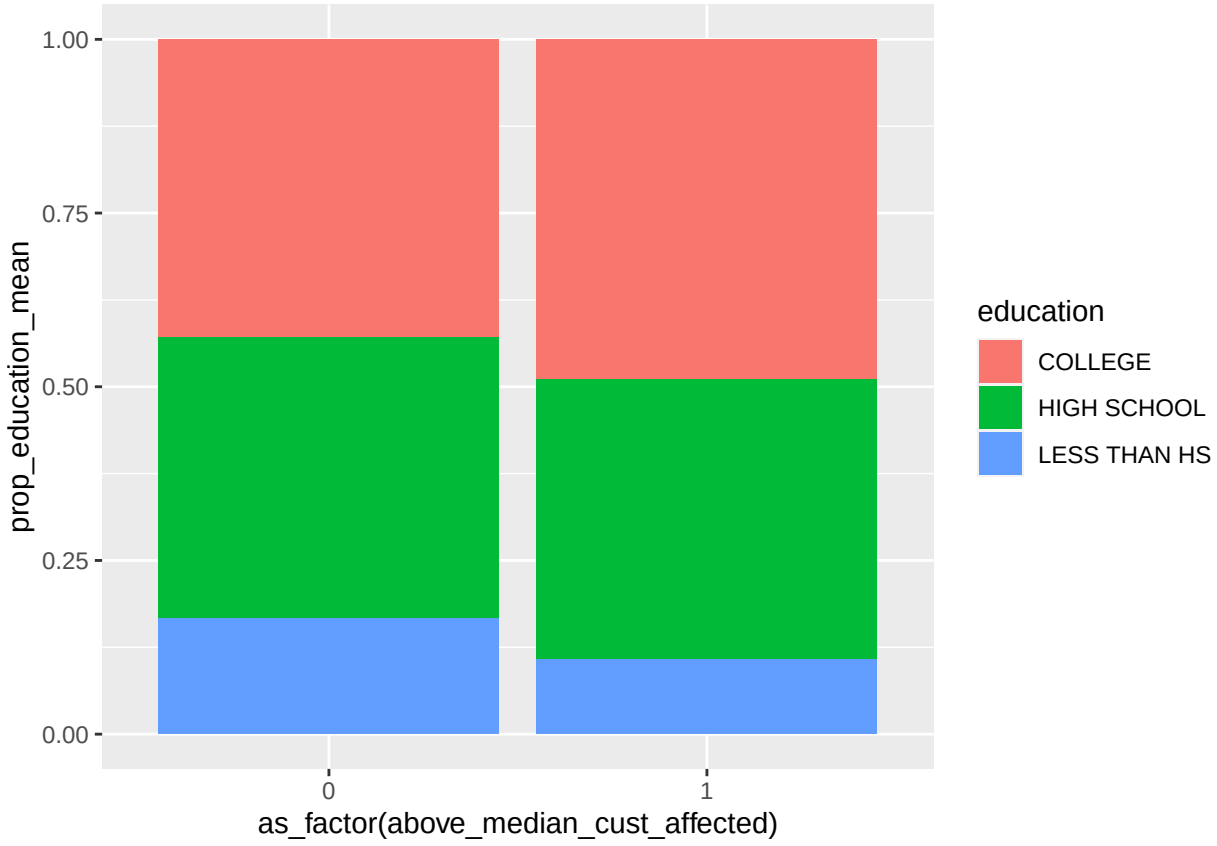



```
acs_outages %>%
  pivot_longer(
    cols = prop_college:prop_less_than_hs,
    names_to = "education",
    values_to = "prop_education"
  ) %>%
  mutate(
    education =
      str_to_title(str_replace_all(str_remove(education, "^prop_"), "_", " "))
  ) %>%
  select(all_of(outcome_vars), education, prop_education) %>%
  ggplot(
    aes(x = median_outage_duration_hr, y = prop_education, color = education)
  ) +
  geom_point(alpha = .4) +
  geom_smooth()
```

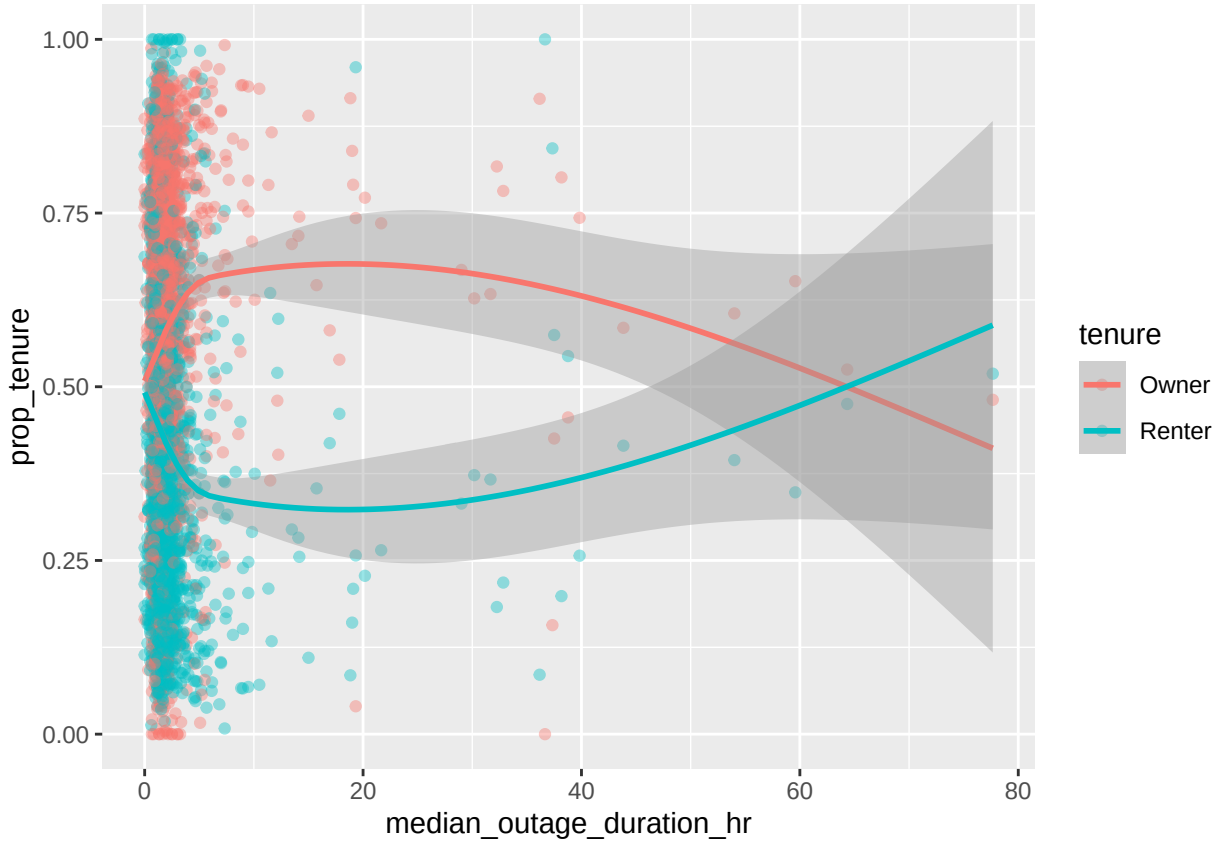


Education

```
acs_outages %>%
  pivot_longer(
    cols = prop_college:prop_less_than_hs,
    names_to = "education",
    values_to = "prop_education"
  ) %>%
  mutate(
    education =
      str_to_upper(str_replace_all(str_remove(education, "^prop_"), "_", " "))
  ) %>%
  select(all_of(outcome_vars), education, prop_education) %>%
  group_by(above_median_cust_affected, education) %>%
  summarise(prop_education_mean = mean(prop_education, na.rm = TRUE)) %>%
  ggplot(
    aes(
      x = as_factor(above_median_cust_affected),
      y = prop_education_mean,
      fill = education
    )
  ) +
  geom_col()
```

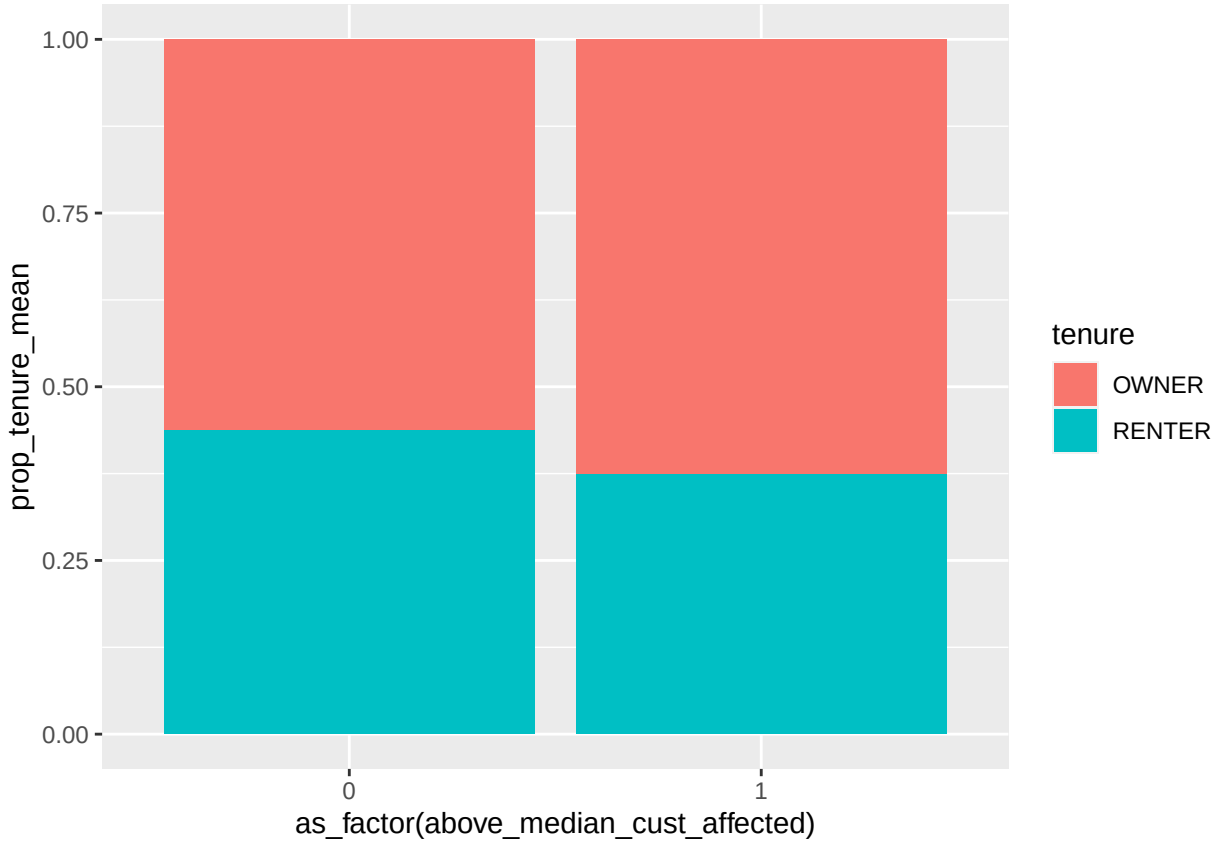



```
acs_outages %>%
  pivot_longer(
    cols = prop_owner:prop_renter,
    names_to = "tenure",
    values_to = "prop_tenure"
  ) %>%
  mutate(
    tenure =
      str_to_title(str_replace_all(str_remove(tenure, "^prop_"), "_", " "))
  ) %>%
  select(all_of(outcome_vars), tenure, prop_tenure) %>%
  ggplot(
    aes(x = median_outage_duration_hr, y = prop_tenure, color = tenure)
  ) +
  geom_point(alpha = .4) +
  geom_smooth()
```

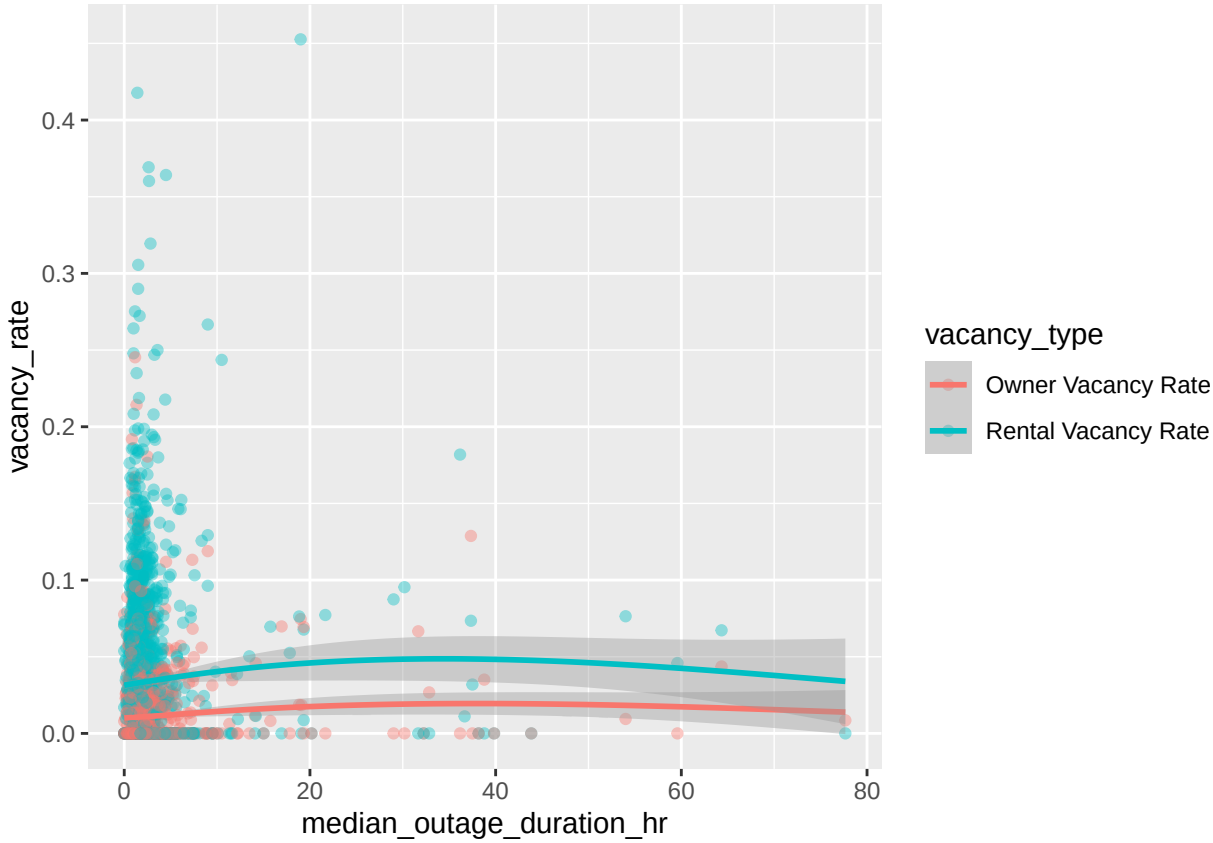


Tenure

```
acs_outages %>%
  pivot_longer(
    cols = prop_owner:prop_renter,
    names_to = "tenure",
    values_to = "prop_tenure"
  ) %>%
  mutate(
    tenure =
      str_to_upper(str_replace_all(str_remove(tenure, "^prop_"), "_", " "))
  ) %>%
  select(all_of(outcome_vars), tenure, prop_tenure) %>%
  group_by(above_median_cust_affected, tenure) %>%
  summarise(prop_tenure_mean = mean(prop_tenure, na.rm = TRUE)) %>%
  ggplot(
    aes(
      x = as_factor(above_median_cust_affected),
      y = prop_tenure_mean,
      fill = tenure
    )
  ) +
  geom_col()
```

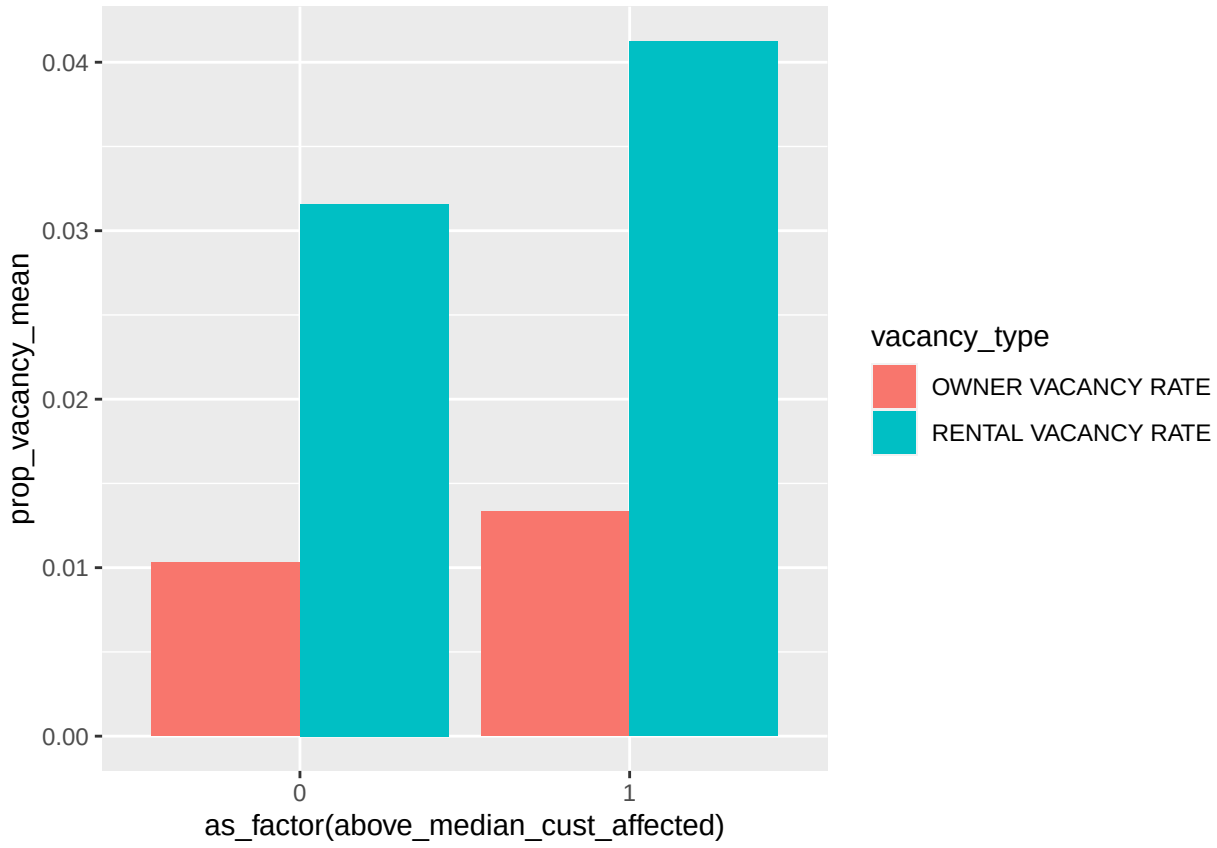



```
acs_outages %>%
  pivot_longer(
    cols = rental_vacancy_rate:owner_vacancy_rate,
    names_to = "vacancy_type",
    values_to = "vacancy_rate"
  ) %>%
  mutate(
    vacancy_type =
      str_to_title(str_replace_all(str_remove(vacancy_type, "^prop_"), "_", " "))
  ) %>%
  select(all_of(outcome_vars), vacancy_type, vacancy_rate) %>%
  ggplot(
    aes(x = median_outage_duration_hr, y = vacancy_rate, color = vacancy_type)
  ) +
  geom_point(alpha = .4) +
  geom_smooth()
```



Vacancy rates

```
acs_outages %>%
  pivot_longer(
    cols = rental_vacancy_rate:owner_vacancy_rate,
    names_to = "vacancy_type",
    values_to = "vacancy_rate"
  ) %>%
  mutate(
    vacancy_type =
      str_to_upper(str_replace_all(str_remove(vacancy_type, "^prop_"), "_", " "))
  ) %>%
  select(all_of(outcome_vars), vacancy_type, vacancy_rate) %>%
  group_by(above_median_cust_affected, vacancy_type) %>%
  summarise(prop_vacancy_mean = mean(vacancy_rate, na.rm = TRUE)) %>%
  ggplot(
    aes(
      x = as_factor(above_median_cust_affected),
      y = prop_vacancy_mean,
      fill = vacancy_type
    )
  ) +
  geom_col(position = "dodge")
```


0.04
0.03
0.02
0.01
0.00
prop_vacancy_mean
0
1
as_factor(above_median_cust_affected)
vacancy_type
OWNER VACANCY RATE
RENTAL VACANCY RATE