9-25-2024 Methods

Lisa Lin

2024-09-25

Data Processing

```
# Get crime data from ArcGIS API and remove (4) entries with missing geometry
# Write cleaned data to GeoJSON file
# crime_dg <- st_read("https://utility.arcgis.com/usrsvcs/servers/3adad6320b7a421bb3826ec8871e2b66/rest/
# crime_dg$Date <- as.Date(fread(".//data//crime_dg.csv")$Date, "%Y/%m/%d")
# crime_dg <- crime_dg[!st_is_empty(crime_dg),]
# st_write(crime_dg, ".//data//crime_dg.geojson")

# Read in Hartford crime and parcel data
# c <- st_read(".//data//crime_dg.geojson")
# p <- st_read(".//data//parcel_hartford.geojson")

# Filter parcels for single family homes only
# Entire apartment or residential complexes are bought less frequently
# and are more complicated to appraise.
# resid_labels <- c("ONE FAMILY") #, "TWO FAMILY", "THREE FAMILY")
# pr <- p[p$State_Use_Description %in% resid_labels, ]

# Filter crimes from 2016-2021
# Property appraisal was made in 2021.
# Assume that about 5 years of crime data is sufficient to capture the effect
# of crime on property values if any effect exists.
# cc <- c[as.Date(c$Date) >= as.Date("2016-01-01") &
#           as.Date(c$Date) <= as.Date("2021-12-31"),]

# Save filtered data
# st_write(pr, ".//data//parcel_hartford_single_family.geojson")
# st_write(cc, ".//data//crime_hartford_2016_2021.geojson")
```

```
# Read in filtered Hartford crime, parcel, and population data
c <- st_read(".//data//crime_hartford_2016_2021.geojson")
```

```
## Reading layer 'crime_hartford_2016_2021' from data source
##   'C:\Users\llint\OneDrive - Yale University\classes\625\CT Property\SDS625\data\crime_hartford_2016_
##   using driver 'GeoJSON'
## Simple feature collection with 197060 features and 12 fields
## Geometry type: POINT
## Dimension:     XY
## Bounding box:  xmin: -72.71865 ymin: 41.72403 xmax: -72.65041 ymax: 41.80719
## Geodetic CRS:  WGS 84
```

```r
p <- st_read(".//data//parcel_hartford_single_family.geojson")
```

```
## Reading layer `parcel_hartford_single_family' from data source
##   `C:\Users\llint\OneDrive - Yale University\classes\625\CT Property\SDS625\data\parcel_hartford_sing
##   using driver `GeoJSON'
## Simple feature collection with 7239 features and 47 fields
## Geometry type: MULTIPOLYGON
## Dimension:     XY
## Bounding box:  xmin: -72.71637 ymin: 41.72374 xmax: -72.65809 ymax: 41.80743
## Geodetic CRS:  WGS 84
```

```r
pop <- read.csv(".//data//population_by_tract.csv", skip = 3, header = T)[,1:3]

# Get polygons for Hartford's census tracts
hartford_tracts <- st_filter(tracts(state = "CT"),
                             subset(county_subdivisions(state = "CT"),
                                    NAMELSAD == "Hartford town"),
                             .predicate = st_within) |>
   st_transform(crs = st_crs(c))
```

```
## Retrieving data for the year 2021
## Retrieving data for the year 2021
```

```r
# Get polygons for Hartford's bodies of water
water <- st_intersection(
  st_transform(area_water("CT", "Hartford"), crs = st_crs(hartford_tracts)),
  hartford_tracts)
```

```
## Retrieving data for the year 2021
```

```
## Warning: attribute variables are assumed to be spatially constant throughout
## all geometries
```

```r
# # Rename tract columns to distinguish from parcel and crime data
# hartford_tracts <- hartford_tracts %>%
#   rename(tract_geometry = geometry, tract_name = NAME, tract_id = GEOID)

# # Extract year from crime date
# c$year <- year(as.Date(c$Date))

# # Join parcel and crime data to census tracts (keep tract geometry)
# tp <- st_join(hartford_tracts, p)
# tc <- st_join(hartford_tracts, c)

# # Extract short tract name from in population data to match tract polygons
# pop$tract_name <- gsub("[^0-9.]", "", pop$Census.Tract)

# # Calculate average housing price by census tract
# ap <- tp %>%
#   group_by(tract_name) %>%
#   summarise(avg_house_value = mean(Assessed_Total, na.rm = TRUE))
```

```r
#
# # Calculate crime rate by census tract and year
# at <- tc[1:1000,] %>%
#   group_by(tract_name, year) %>%
#   summarise(crime_count = n()) %>%
#   left_join(pop, by = "tract_name") %>%
#   mutate(crime_rate_per_1000 = crime_count / Estimated.Population * 1000) %>%
#   select(tract_name, year, crime_rate_per_1000) %>%
#   pivot_wider(names_from = year, values_from = crime_rate_per_1000) %>%
#   left_join(ap, by = "tract_name") %>%
#   select(tract_name, everything())
#
#   write.csv(".//data//crime_rate_by_tract.csv", row.names = FALSE)
#
# tc
```

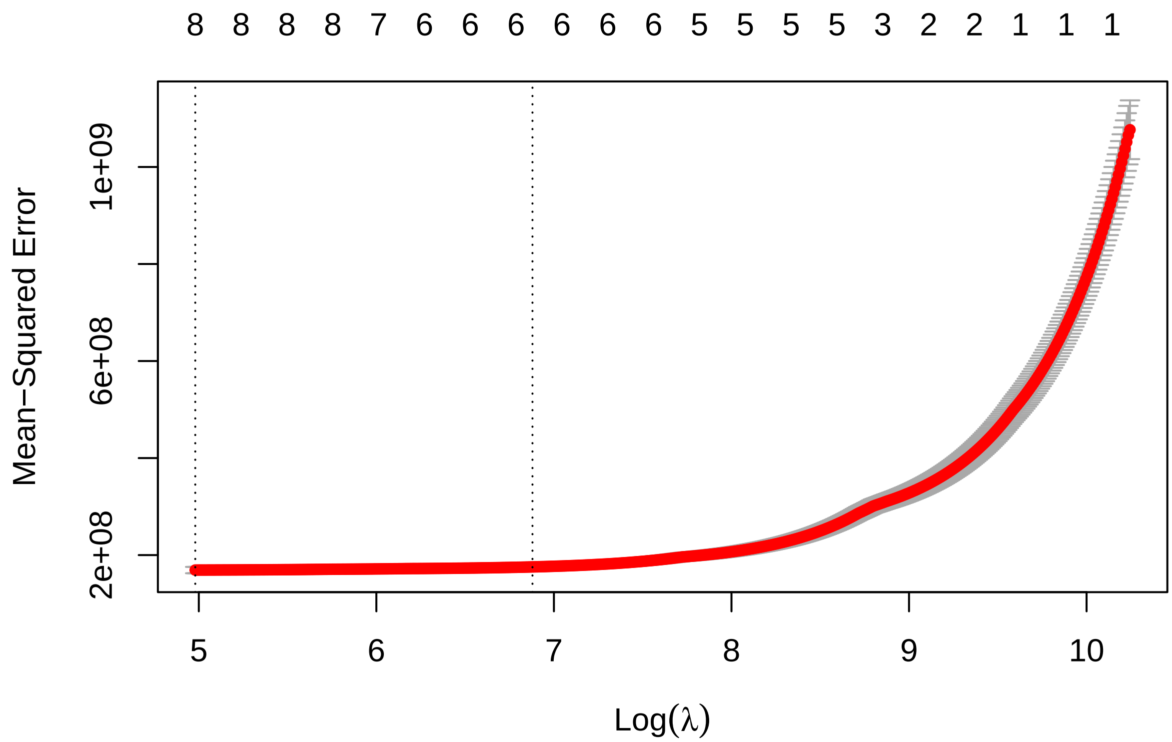
Analysis

```r
library(glmnet)
```

```
## Loading required package: Matrix

##
## Attaching package: 'Matrix'

## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack

## Loaded glmnet 4.1-8
```

```r
# Filter major property crimes and violent crimes
stealin <- c[grep("ROBBERY|BURGLARY|LARCENY|THEFT|STOLEN",
                  c$UCR_1_Category), ]
hurtin <- c[grep("ASSAULT|HOMICIDE|SHOOTING", c$UCR_1_Category), ]

# Get number of thefts and violent crimes within 0.1 miles of each parcel
p$thefts <- sapply(
  st_is_within_distance(p, stealin, dist = units::set_units(0.1, "mi")),
  length)
p$violence <- sapply(
  st_is_within_distance(p, hurtin, dist = units::set_units(0.1, "mi")),
  length)
# Re-order condition description of parcels
p$Condition_Description <- ordered(
  p$Condition_Description,
  levels = c("Dilapidated", "Very Poor", "Poor", "Fair",
             "Fair-Avg", "Average", "Avg-Good", "Good",
             "Good-VG", "Very Good", "Excellent"))
```

```r
# Format predictors and response variable from parcel data
X <- p %>%
  dplyr::select(Condition_Description,
                AYB, Living_Area, Effective_Area,
                Total_Rooms, Number_of_Bedroom, Number_of_Baths,
                thefts, violence) %>%
  # mutate(sqrt_Living_Area = sqrt(Living_Area),
  #        sqrt_Effective_Area = sqrt(Effective_Area),
  #        thefts1 = thefts + 1,
  #        thefts2 = thefts*2) %>%
  sf::st_drop_geometry()
X <- data.matrix(X)
y <- p$Assessed_Total

# Remove rows with missing values
w <- complete.cases(X, y)
X <- X[w, ]
y <- y[w]

# Select variables
# Run lasso regression
cv_tune.lasso_model = suppressMessages(suppressWarnings(
  cv.glmnet(x = X,
            y = y,
            nlambda = 1000,
            nfolds = 500,
            pmax = 15,
            parallel = TRUE)))

plot(cv_tune.lasso_model)
```




```
lasso_modelmin = glmnet(x = X, y = y, lambda = cv_tune.lasso_model$lambda.min)
lasso_model1se = glmnet(x = X, y = y, lambda = cv_tune.lasso_model$lambda.1se)

# Does not select Total_Rooms
coef(lasso_modelmin)
```

```
## 10 x 1 sparse Matrix of class "dgCMatrix"
##                                   s0
## (Intercept)            -257670.52057
## Condition_Description     4050.10033
## AYB                        124.69109
## Living_Area                 26.20137
## Effective_Area              13.40379
## Total_Rooms                   .
## Number_of_Bedroom        -1082.48428
## Number_of_Baths           5242.56708
## thefts                      26.63910
## violence                  -232.04680
```

```
# Does not select Total_Rooms, Number_of_Bedroom, thefts
coef(lasso_model1se)
```

```
## 10 x 1 sparse Matrix of class "dgCMatrix"
##                                   s0
## (Intercept)            -154584.83105
```

```
## Condition_Description    3358.89052
## AYB                        74.69969
## Living_Area                24.53254
## Effective_Area             12.79626
## Total_Rooms                 .
## Number_of_Bedroom           .
## Number_of_Baths          4623.92283
## thefts                      .
## violence                 -179.85860
```

```
vars_keepmin = rownames(coef(lasso_modelmin))[
  which(as.matrix(coef(lasso_modelmin)) != 0)][-1]
vars_keep1se = rownames(coef(lasso_model1se))[
  which(as.matrix(coef(lasso_model1se)) != 0)][-1]

# Fit linear models with selected variables
m_min <- lm(log(Assessed_Total) ~ thefts + violence
           + sqrt(Living_Area) + sqrt(Effective_Area)
           + AYB + Number_of_Bedroom + Number_of_Baths
           + Condition_Description, data = p)
m_1se <- lm(log(Assessed_Total) ~ violence
           + sqrt(Living_Area) + sqrt(Effective_Area)
           + AYB + Number_of_Baths
           + Condition_Description, data = p)
summary(m_min)
```

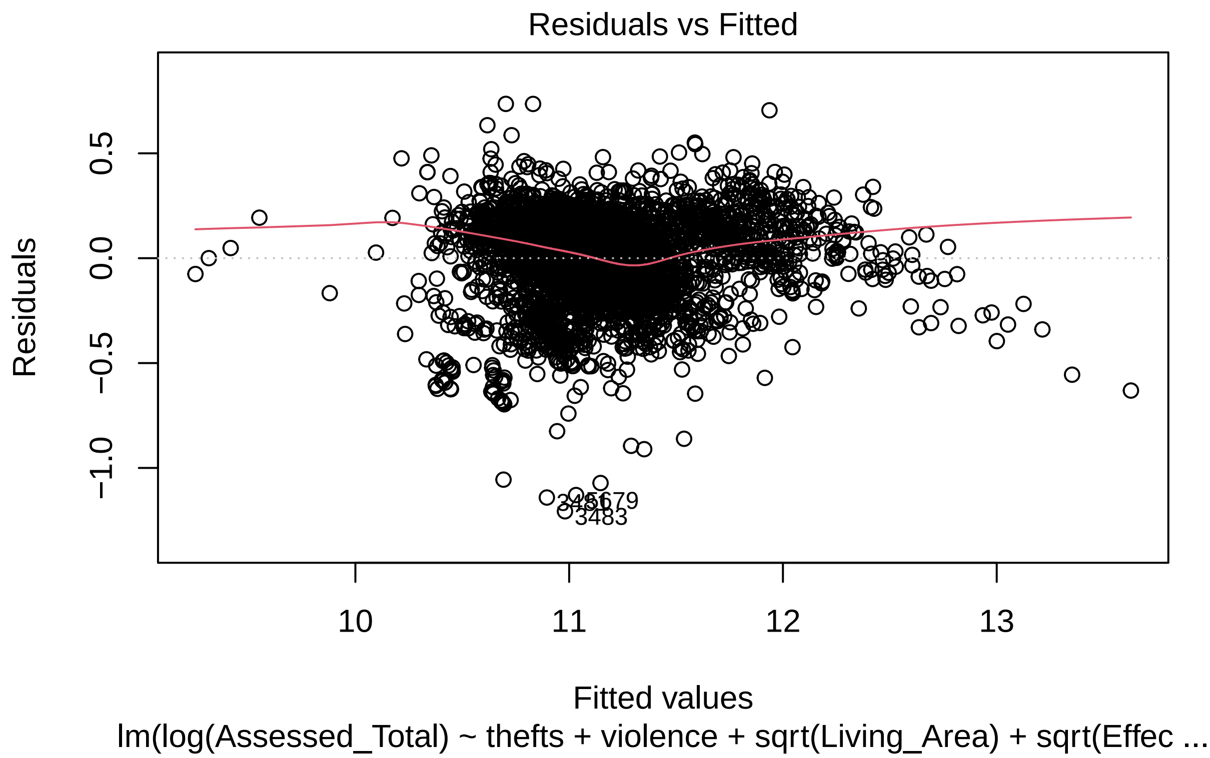
```
##
## Call:
## lm(formula = log(Assessed_Total) ~ thefts + violence + sqrt(Living_Area) +
##     sqrt(Effective_Area) + AYB + Number_of_Bedroom + Number_of_Baths +
##     Condition_Description, data = p)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.20660 -0.06559  0.01653  0.08745  0.73563
##
## Coefficients:
##                            Estimate Std. Error t value Pr(>|t|)
## (Intercept)               6.609e+00  1.555e-01  42.494  < 2e-16 ***
## thefts                    1.006e-03  8.077e-05  12.459  < 2e-16 ***
## violence                 -3.908e-03  1.246e-04 -31.369  < 2e-16 ***
## sqrt(Living_Area)         2.351e-02  3.964e-04  59.298  < 2e-16 ***
## sqrt(Effective_Area)      1.065e-02  3.219e-04  33.077  < 2e-16 ***
## AYB                       1.508e-03  7.783e-05  19.382  < 2e-16 ***
## Number_of_Bedroom        -7.861e-03  2.668e-03  -2.947 0.003222 **
## Number_of_Baths           5.053e-02  3.929e-03  12.859  < 2e-16 ***
## Condition_Description.L   1.439e+00  4.482e-02  32.099  < 2e-16 ***
## Condition_Description.Q  -6.056e-01  4.011e-02 -15.100  < 2e-16 ***
## Condition_Description.C   2.630e-01  3.634e-02   7.237 5.07e-13 ***
## Condition_Description^4  -1.782e-01  3.788e-02  -4.704 2.60e-06 ***
## Condition_Description^5   2.125e-01  3.883e-02   5.472 4.60e-08 ***
## Condition_Description^6  -9.673e-02  3.546e-02  -2.728 0.006383 **
## Condition_Description^7  -9.615e-03  2.918e-02  -0.330 0.741741
## Condition_Description^8   9.650e-04  2.199e-02   0.044 0.964999
```

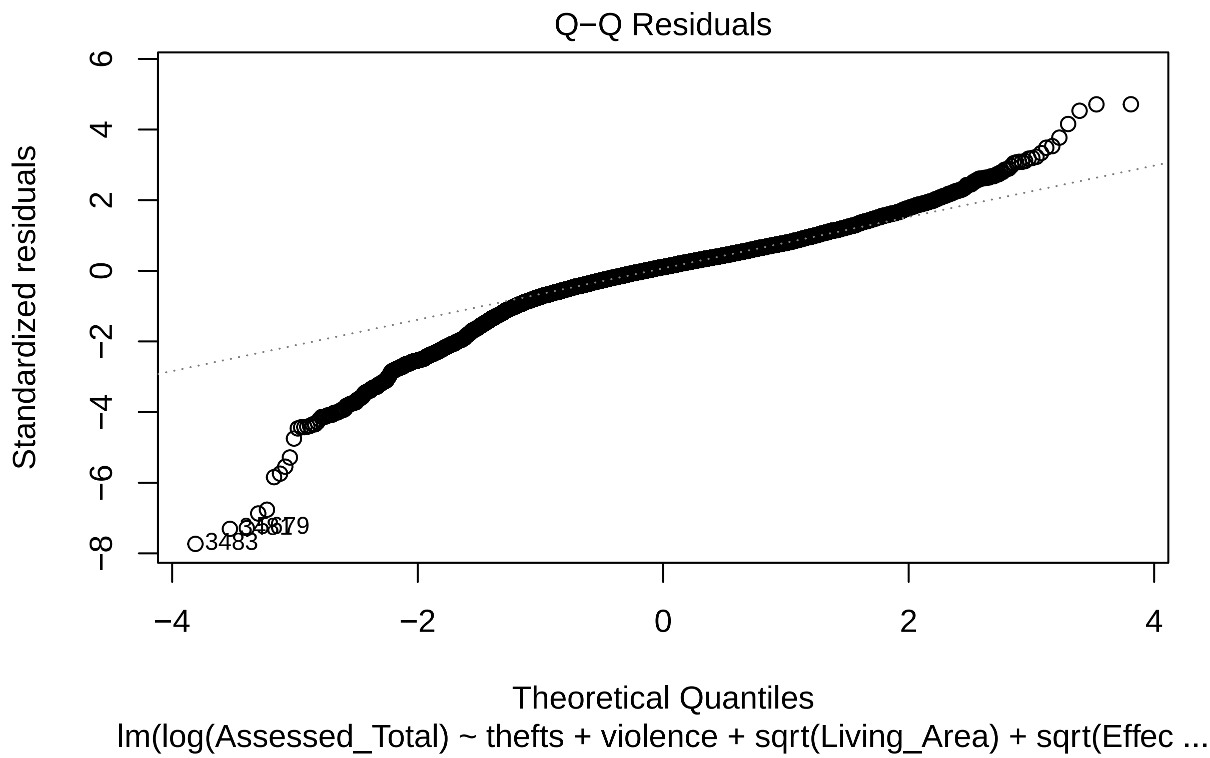
```
## Condition_Description^9   5.529e-02  1.598e-02   3.460 0.000543 ***
## Condition_Description^10 -3.143e-02  1.004e-02  -3.130 0.001755 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1562 on 7202 degrees of freedom
##   (19 observations deleted due to missingness)
## Multiple R-squared:  0.7871, Adjusted R-squared:  0.7866
## F-statistic:  1566 on 17 and 7202 DF,  p-value: < 2.2e-16
```

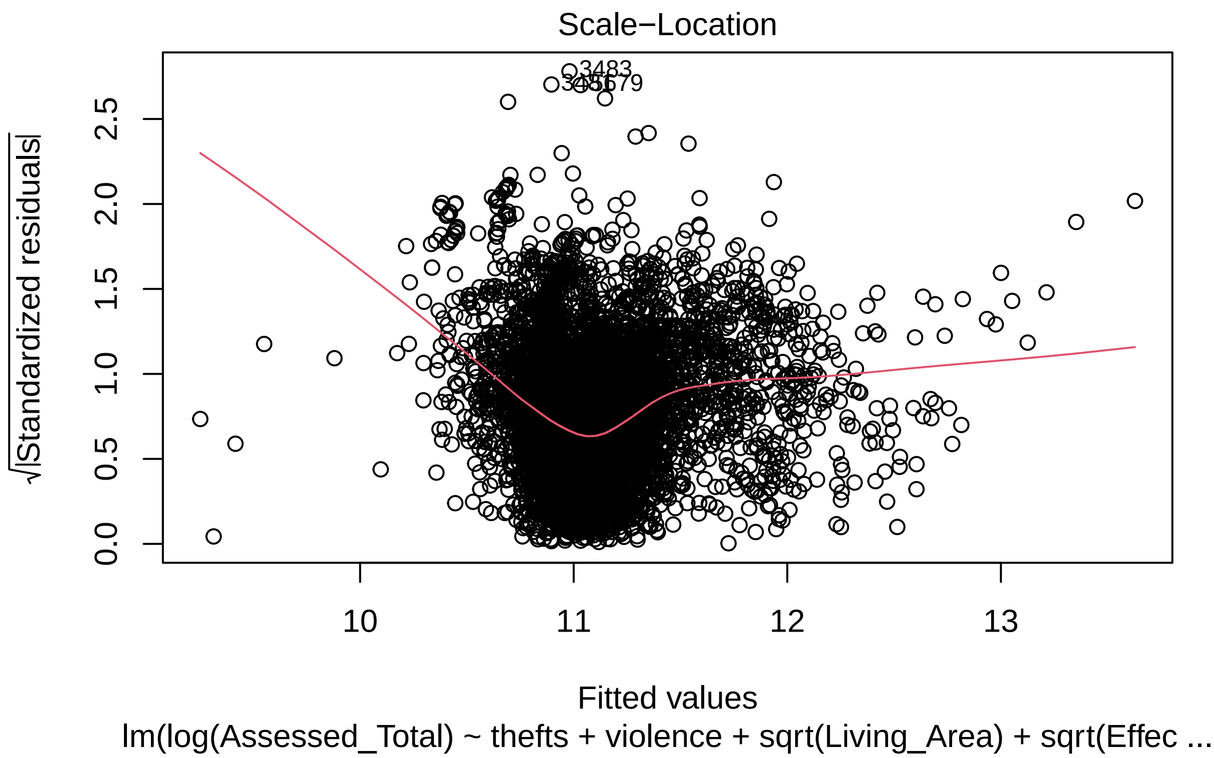
```r
summary(m_1se)
```

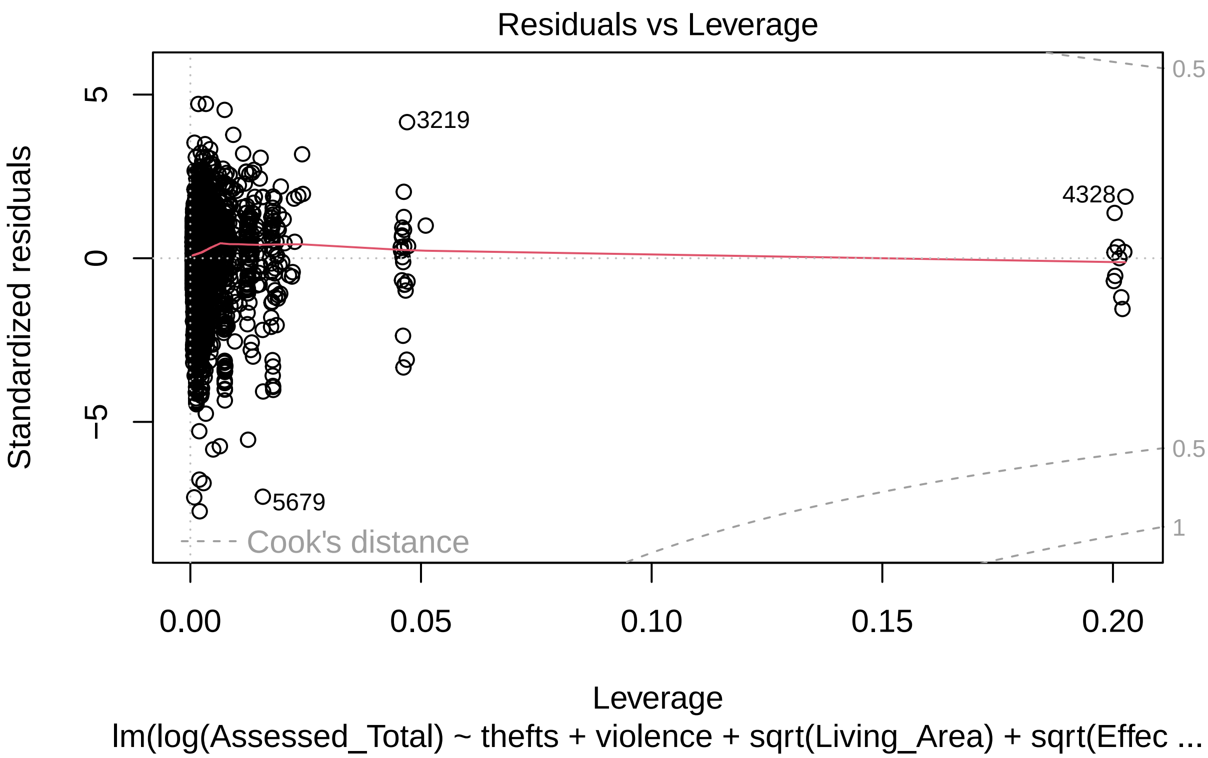
```
##
## Call:
## lm(formula = log(Assessed_Total) ~ violence + sqrt(Living_Area) +
##     sqrt(Effective_Area) + AYB + Number_of_Baths + Condition_Description,
##     data = p)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.20134 -0.06853  0.01696  0.09080  0.75484
##
## Coefficients:
##                            Estimate Std. Error t value Pr(>|t|)
## (Intercept)               6.914e+00  1.550e-01  44.606  < 2e-16 ***
## violence                 -2.658e-03  7.160e-05 -37.126  < 2e-16 ***
## sqrt(Living_Area)         2.302e-02  3.464e-04  66.440  < 2e-16 ***
## sqrt(Effective_Area)      1.065e-02  3.253e-04  32.746  < 2e-16 ***
## AYB                       1.363e-03  7.766e-05  17.547  < 2e-16 ***
## Number_of_Baths           4.718e-02  3.946e-03  11.957  < 2e-16 ***
## Condition_Description.L   1.422e+00  4.531e-02  31.377  < 2e-16 ***
## Condition_Description.Q  -5.995e-01  4.055e-02 -14.783  < 2e-16 ***
## Condition_Description.C   2.531e-01  3.675e-02   6.888 6.16e-12 ***
## Condition_Description^4  -1.719e-01  3.831e-02  -4.486 7.38e-06 ***
## Condition_Description^5   2.138e-01  3.928e-02   5.444 5.38e-08 ***
## Condition_Description^6  -9.869e-02  3.586e-02  -2.752 0.005937 **
## Condition_Description^7  -8.460e-03  2.951e-02  -0.287 0.774352
## Condition_Description^8  -2.037e-04  2.224e-02  -0.009 0.992693
## Condition_Description^9   5.585e-02  1.616e-02   3.455 0.000553 ***
## Condition_Description^10 -3.370e-02  1.015e-02  -3.319 0.000906 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.158 on 7204 degrees of freedom
##   (19 observations deleted due to missingness)
## Multiple R-squared:  0.7821, Adjusted R-squared:  0.7817
## F-statistic:  1724 on 15 and 7204 DF,  p-value: < 2.2e-16
```

```r
plot(m_min)
```


Residuals vs Fitted
Residuals
0.5
0.0
−0.5
−1.0
10
11
12
13
Fitted values
lm(log(Assessed_Total) ~ thefts + violence + sqrt(Living_Area) + sqrt(Effec ...
3483

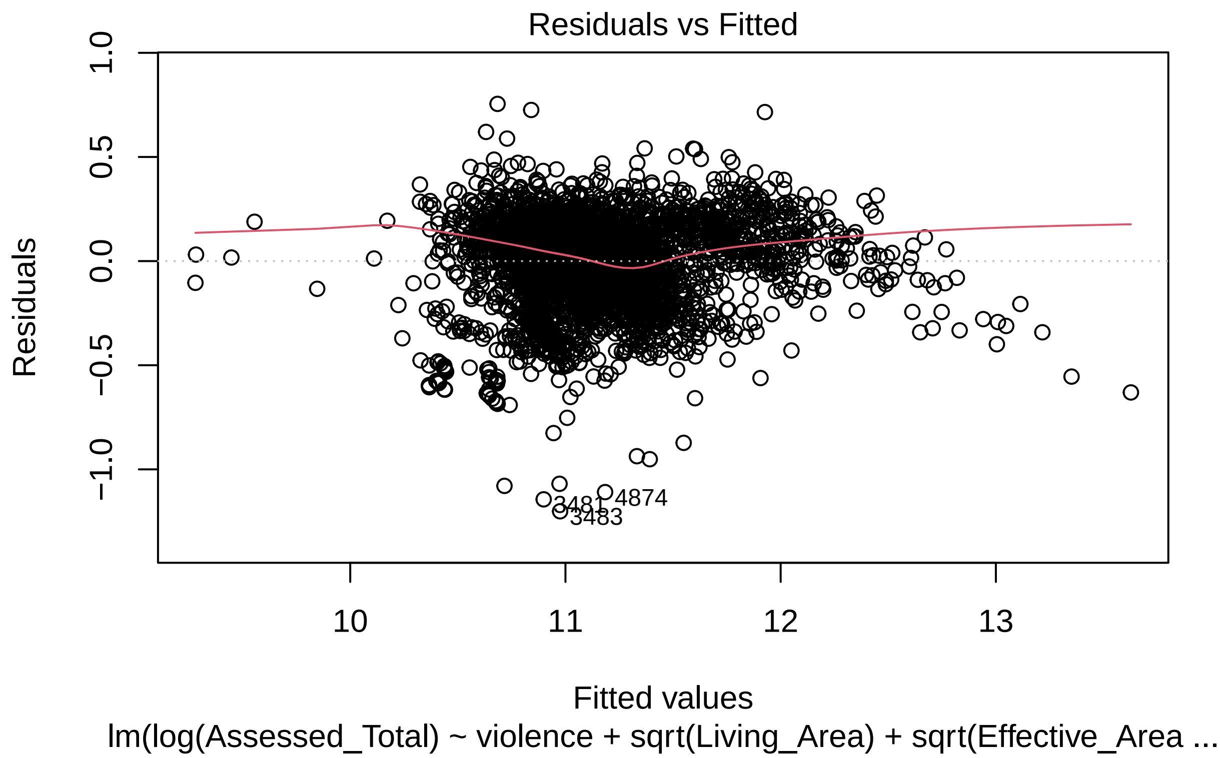

Q−Q Residuals
Standardized residuals
Theoretical Quantiles
lm(log(Assessed_Total) ~ thefts + violence + sqrt(Living_Area) + sqrt(Effec ...
3483
3481
3479

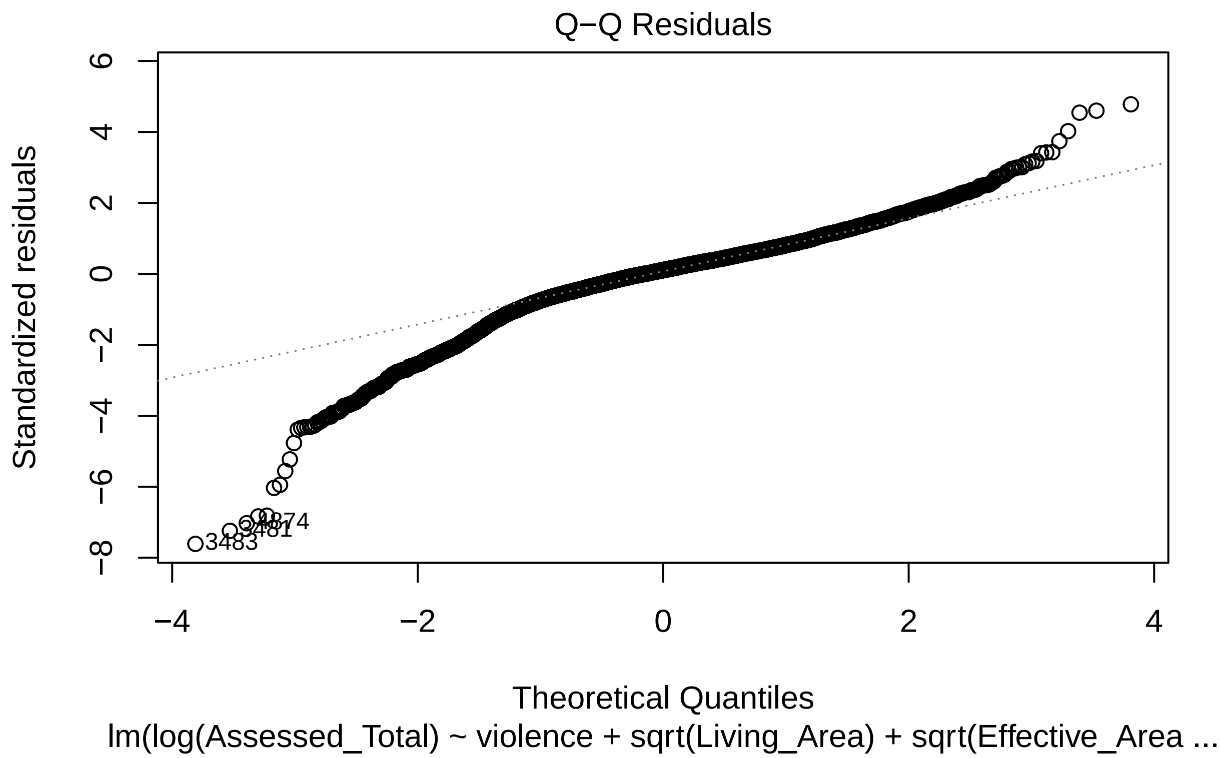

Scale−Location
3483
√|Standardized residuals|
0.0
0.5
1.0
1.5
2.0
2.5
10
11
12
13
Fitted values
lm(log(Assessed_Total) ~ thefts + violence + sqrt(Living_Area) + sqrt(Effec ...

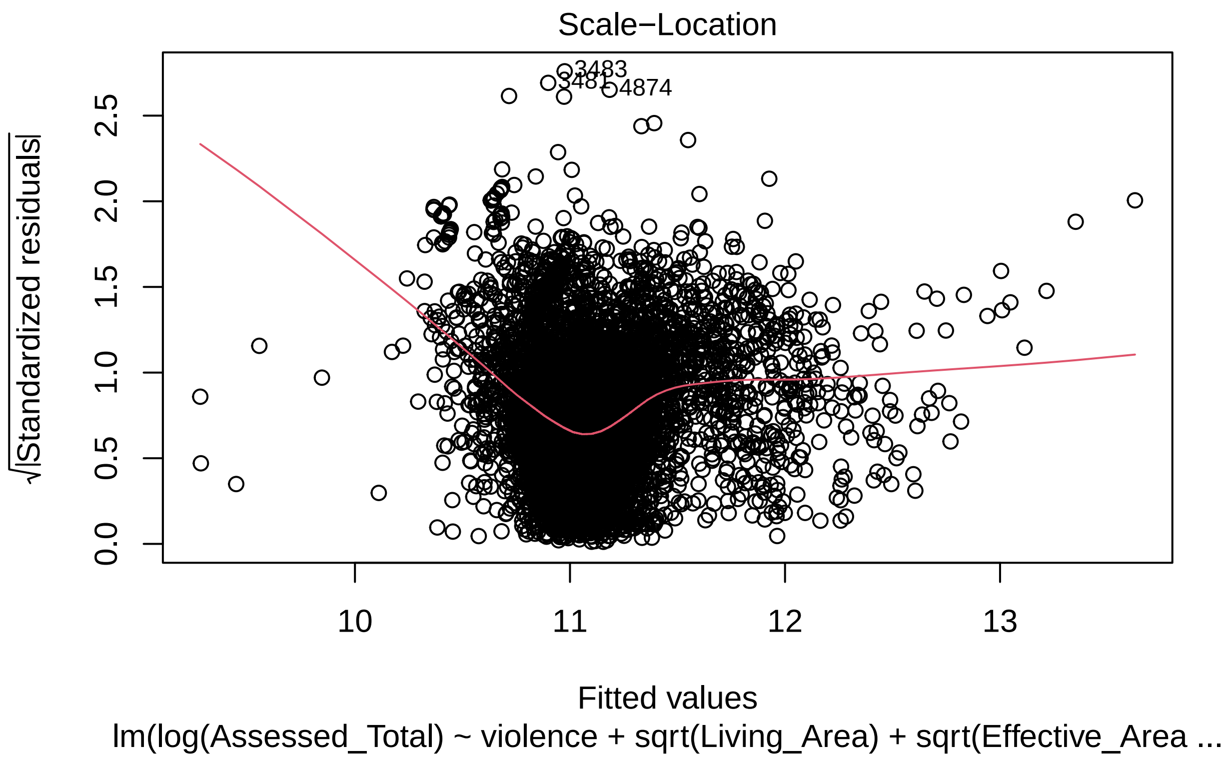


Residuals vs Leverage
Standardized residuals
5
0
−5
3219
4328
5679
Cook's distance
0.5
0.5
1
0.00
0.05
0.10
0.15
0.20
Leverage
lm(log(Assessed_Total) ~ thefts + violence + sqrt(Living_Area) + sqrt(Effec ...

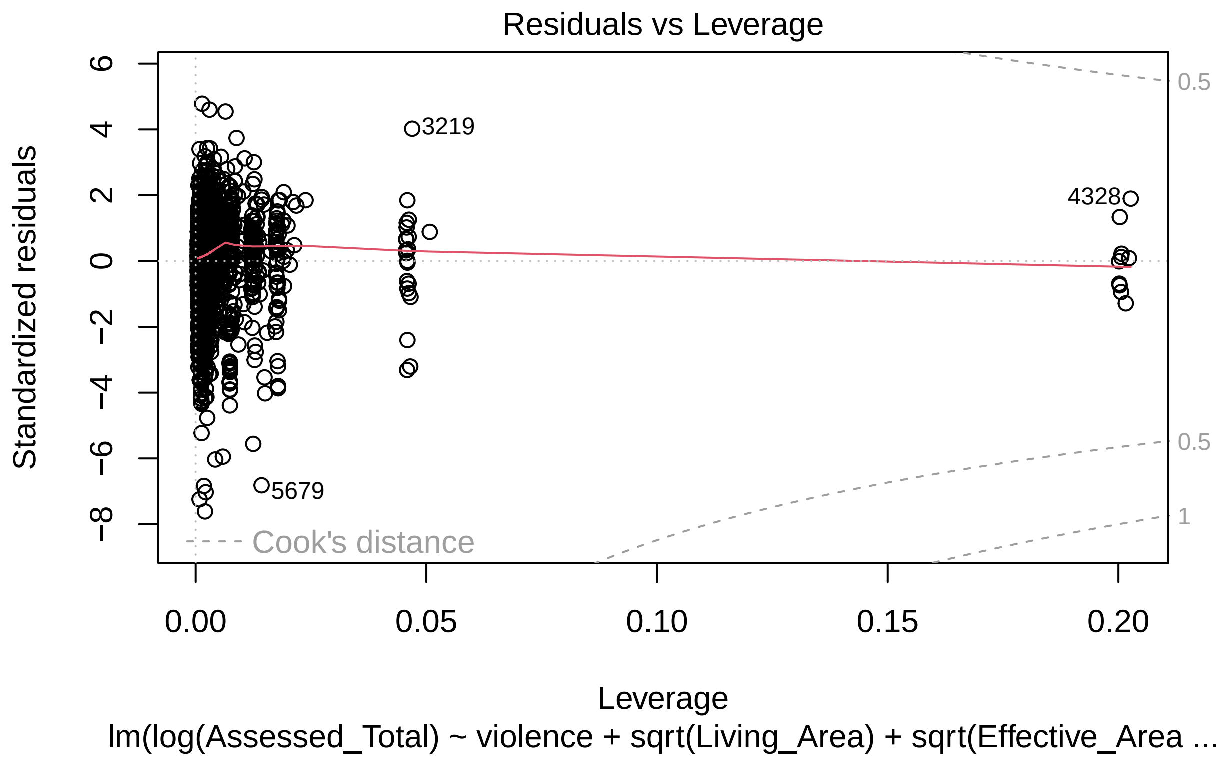

```
plot(m_1se)
```


Residuals vs Fitted
Residuals
1.0
0.5
0.0
−0.5
−1.0
10
11
12
13
Fitted values
lm(log(Assessed_Total) ~ violence + sqrt(Living_Area) + sqrt(Effective_Area ...
3481
3483
4874


Q−Q Residuals
Standardized residuals
Theoretical Quantiles
lm(log(Assessed_Total) ~ violence + sqrt(Living_Area) + sqrt(Effective_Area ...
3483
3481
4874
−8
−6
−4
−2
0
2
4
6
−4
−2
0
2
4



Scale−Location
√|Standardized residuals|
0.0
0.5
1.0
1.5
2.0
2.5
3483
3481
4874
10
11
12
13
Fitted values
lm(log(Assessed_Total) ~ violence + sqrt(Living_Area) + sqrt(Effective_Area ...



Other stuff

```
# # Calculate centroids of parcels and crime
# centroids <- st_centroid(p)
# coords <- st_coordinates(centroids)
# p$centroid.x <- coords[, 'X']
# p$centroid.y <- coords[, 'Y']
#
# centroids <- st_centroid(c)
# coords <- st_coordinates(centroids)
# c$centroid.x <- coords[, 'X']
# c$centroid.y <- coords[, 'Y']
#
#
# # Filter property crimes and violent crimes
# stealin <- c[grep("ROBBERY|BURGLARY|LARCENY|THEFT|STOLEN",
#                   c$UCR_1_Category), ]
# hurtin <- c[grep("ASSAULT|HOMICIDE|SHOOTING", c$UCR_1_Category), ]
#
# # Get number of thefts and violent crimes within 0.1 miles of each parcel
# p$thefts <- sapply(
#   st_is_within_distance(p, stealin, dist = units::set_units(0.1, "mi")),
#   length)
# p$violence <- sapply(
```

```
#   st_is_within_distance(p, hurtin, dist = units::set_units(0.1, "mi")),
#   length)
#
# st_within(p$geometry[1], p$geometry[1])
# st_within(p$geometry[1], hartford_tracts[1])
```

```
# # For each census tract, get average property value in 2021 and
# # average annual crime count
# # Create a new dataframe
# library(dplyr)
# # Add a column for year
# c <- c %>%
#   mutate(year = year(as.Date(Date)))
# # Drop geometry data
# cc <- sf::st_drop_geometry(c)
# pp <- sf::st_drop_geometry(p)
#
# # Calculate total crimes per year
# crimes_per_year <- table(cc$year)
#
#
# select(year) %>%
#   group_by(year) %>%
#   summarise(total_crime_per_yr = n())
#
#
# # Aggregate crime to census level to get crime rate
```

```
# # Plot census tracts colored by crime rates and sized by property values
# l1 = leaflet(c_tract) %>%
#
#   addProviderTiles('CartoDB.Positron') %>%
#
#   ## census tracts
#   addPolygons(fillColor = ~pal1(rescaled.house.value),
#               label = ~label %>% lapply(htmltools::HTML),
#               weight = 0.5,
#               color = 'black',
#               fillOpacity = 0.8) %>%
#
#   # stops
#   addCircleMarkers(data = ds,
#                    lng = ~stop1_lon,
#                    lat = ~stop1_lat,
#                    label = ~label %>% lapply(htmltools::HTML),
#                    color = pubred,
#                    radius = ~log(`n_to_Grand Central` + `n_to_New Haven`)) %>%
#
#   setView(lng = -73.3,
```

```
#           lat = 41.21979,
#           zoom = 10) %>%
#
#   addTiles()
```

```
# # Plot sample of crimes and parcels
#
# # library(mapview)
# # mapview(dplyr::sample_n(c, 1e3), dplyr::sample_n(p, 1e3))
#
# g <- ggplot(dplyr::sample_n(c, 1e3)) +
#    geom_sf(data = hartford_tracts) +
#    geom_sf(data = dplyr::sample_n(p, 1e3)) +
#    geom_density_2d(aes(X,Y), data = ~cbind(.x, st_coordinates(.x))) +
#    stat_sf_coordinates(size = 0.1, color = "red") +
#    labs(x = "Latitude", y = "Longitude") +
#    theme_bw()
#
# p <- toWebGL(ggplotly(g))
# p$x$data[[4]]$hoverinfo <- "none"
# p
```

```
# # Plot average residential parcel value by census tract
# # with crime counts binned by area
#
# # Map parcel address to
#
#
# # Get average residential parcel value by census tract
# parcel_dg$Zone
#
# hartford_tracts
#
# parcel_dg$avg_residential_value <- parcel_dg$AV_LAND / parcel_dg$AV_TOTAL
```

Data Exploration

References

Spatial regression

https://oerstatistics.wordpress.com/wp-content/uploads/2016/03/intro_to_r.pdf#page=68.08

https://crd230.github.io/lab8.html

Kernel density estimation

https://seeing-statistics.com/issue4/