Lab 4: Regression Competition

One last time through the hot New Jersey night…

Paul Nguyen

group_F_process

We eliminate undesired variables and also automatically generate a new dataframe with all the variables squared and cubic(ed).

group_F_automatic

We run regsubsets on the dataframe and generate a lm object automatically, no need of manually checking and writing the LM.

```r
library(tidyverse)
library(leaps)

d <- read.csv("http://andrewpbray.github.io/data/crime-train.csv")

# Data wrangling
group_F_process <- function(training_data) {
  dw<-as_tibble(training_data)
  dw<-select (training_data,-c(state,county,community,communityname,LemasSwornFT,LemasSwFTPerPop,LemasSw
  vars<-c()
  for (i in 1:(length(dw)-1)) {
    vars<- c(vars, names(dw)[i])
  }
  sqrd<-data.frame(lapply(vars, function(x){dw[,x]^(1/2)}))
  cubc<-data.frame(lapply(vars, function(x){dw[,x]^(1/3)}))
  names(sqrd)<-paste0(vars, "Sq")
  names(cubc)<-paste0(vars, "Cub")
  dw<-cbind(dw, sqrd,cubc)
  return(dw)
}

# Manually fits model
group_F_fit <- function(training_data) {
  m1 <- lm(ViolentCrimesPerPop~
             MalePctDivorceCub+
             PctKids2ParCub+
             PctIlleg+
             PctPersDenseHousSq+
             RentLowQ+
             MedRent,
           training_data)
  m1
}

# Gets MSE
group_F_MSE <- function(model, data) {
  mean((data$ViolentCrimesPerPop - predict.lm(model, data)) ^ 2)
}

# Automatically fits model
group_F_automated_fit <- function(data, met) {
  leaps<-regsubsets(ViolentCrimesPerPop~.,
                    data = data,
                    nvmax = 70,
                    method = met)
  best<-summary(leaps)$which[which.max(summary(leaps)$adjr2),]
  variables <- c()
  for (i in 2:length(best)) {
    if (best[i] == TRUE) {
      variables <- c(variables, names(best)[i])
    }
  }
  vars<- paste(variables, collapse = "+")
  formula<- paste("lm(ViolentCrimesPerPop ~ ",vars,", data =", deparse(substitute(data)),")")
  m1<-eval(parse(text=formula))
  return(m1)
}

dw <- group_F_process(d)
bestF <- group_F_automated_fit(dw, "forward")
```

```
## Warning in leaps.setup(x, y, wt = wt, nbest = nbest, nvmax = nvmax,
## force.in = force.in, : 1 linear dependencies found
```

```
## Reordering variables and trying again:
```

```r
bestB <- group_F_automated_fit(dw, "backward")
```

```
## Warning in leaps.setup(x, y, wt = wt, nbest = nbest, nvmax = nvmax,
## force.in = force.in, : 1 linear dependencies found
```

```
## Reordering variables and trying again:
```

```r
MSE_F <- group_F_MSE(bestF, dw)
MSE_B <- group_F_MSE(bestB, dw)
```

looks like backward is the better way to go

Problem Set

2.

a. The lasso, relative to least squares, is less flexible so that it it reduces bias when there is collinearity in the data. It will give improved prediction accuracy when its increase in bias is less that its decrease in variance

b. Similarly to the lasso, the ridge regression is less flexible than the least squares method so that it provides more stability to the model. It will give improved predictino accuracy when its increase in bias is less thatn its decrease in variance.

3. Suppose we estimate the regression coefficients in a linear regression model by minimizing $\sum_{i=1}^{n}(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij})^2$ subject to $\sum_{j=1}^{p}|\beta_j| \leq s$

(a) Note that increasing s means that our coefficients become larger and larger… they are less constrained. This means that the model will fit to the training data increasingly well. thus, as we increase s from 0, the training RSS will steadily decrease.

(b) I think that a similar reasoning can be applied to Test RSS. At $s = 0$, the model is just $B_0 =$ mean of the data. As we increase s, the model becomes more and more complex, until s becomes large enough that it encompasses $\hat{\beta}$ where $\hat{\beta}$ is the least sum of squares model. If our test data is simlar to our training data, test RSS should decrease as we increase s. However, I believe that the model may become too complex, and so Test RSS may eventually start decreasing when we start overfitting our data.

(c) For variance, we can think about this as model complexity. When $s = 0$, there is no variance. The beta terms are just 0. As we increase s (this is equivalent to decreasing λ in the lasso function), the beta terms start emerging, and variance should start to steadily increase.

(d) For squared bias, when $s = 0$, bias here is super big. Then when we increase s, we start to add model complexity and so bias just starts to decrease.

(e) I think here irreducible error remains constant. S really only affects the beta terms, which estimate y. However, irreducible error comes from the data not being perfect and having a little bit of jitter no matter what level we set s to.

4. Suppose we estimate the regression coefficients in a linear regression model by minimizing $\sum_{i=1}^{n}(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij})^2 + \lambda \sum_{j=1}^{p} \beta_j^2$

(a) Here, this is a ridge regression whose complexity starts at the least squares method and then decreases as λ increases. Thus, when we increase λ from 0, the training RSS will decrease pretty steadily. The training RSS would have been the highest at $\lambda = 0$.

(b) When we repeat for test RSS, I believe that the test RSS will decrease and then eventually start increasing. My reasoning: At $\lambda = 0$, this is equivalent to the least squares solution.. it may be needlessly complex. When we increase lambda, we start punishing for needless predictors, which will bring our test RSS down. However, when lambda becomes too high, then having any predictors at all will punish our model, and then our bias is overpowering any decreases in variance that we brought.

(c) Increasing lambda decreases our model complexity. Thus, our variance will steadily decrease.

(d) Squared Bias will steadily increase as we increase lambda. We decrease model complexity in exchange for introducing bias to our model.

(e) Again, irreducible error will remain constant. It is not affected by our model predictors.

5. given: $n = 2, p = 2, x_{11} = x_{12}$ suppose that $y_1 + y_2 = 0, x_{12} + x_{21} = 0$ so that the estimate for the intercept for a ridge regression, lasso, or least squares is 0. $\hat{\beta}_0 = 0$

(a) for ridge regression, minimize $\sum_{j=1}^{p}(y_i - \beta_j)^2 + \lambda \sum_{j=1}^{p} \beta_j^2$ $((y_1 - \beta_1)^2 + (y_2 - \beta_2)^2) + ((\lambda\beta_1)^2 + (\lambda\beta_2)^2)$

(b) in this setting, ridge regression estimates take the form: $\hat{\beta}_j^R = y_j/(1 + \lambda)$ (note talk to andrew about this)

(c) for lasso, minimize $\sum_{j=1}^{p}(y_i - \beta_j)^2 + \lambda \sum_{j=1}^{p} |\beta_j|$ $((y_1 - \beta_1)^2 + (y_2 - \beta_2)^2) + (\lambda|\beta_1| + \lambda|\beta_2|)$

(d) lasso estimates:

$$\hat{\beta}_2 = y_2 - \lambda/2 \implies \hat{\beta}_2 = -y_1 - \lambda/2$$

$$\hat{\beta}_1 \neq \hat{\beta}_2$$

$$y_1 > \lambda/2 \implies y_2 < \lambda/2$$

$$\hat{\beta}_1 = y_1 - \lambda/2$$

$$\hat{\beta}_2 = y_2 + \lambda/2 \implies \hat{\beta}_2 = -y_1 + \lambda/2$$

Again,

$$\hat{\beta}_1 \neq \hat{\beta}_2$$

6.

(a) For some y_1 and $\lambda > 0$, plot 6.12 as a function of β_1 Let $y_1 = 2$ and $\lambda = 1$

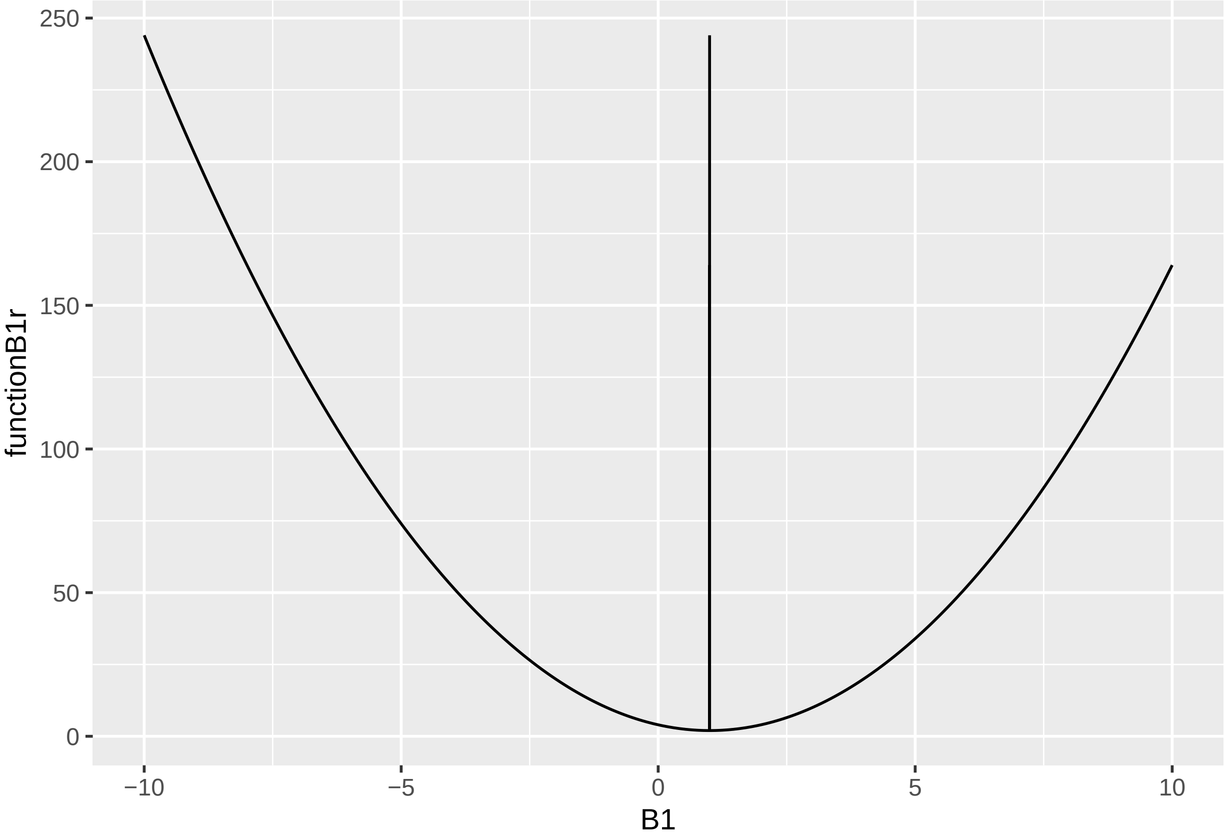
```
y1 <- 2
lambda <-1
B1 <- seq(from = -10, to = 10, by = .01)
functionB1r <- (y1)^2 -2*y1*B1  + B1^2+ lambda*B1^2
functionB1r2 <- (y1-B1)^2 + lambda*(B1^2)
min(functionB1r)
```

```
## [1] 2
```

```
min(functionB1r2)
```

```
## [1] 2
```

```
df <- data.frame(B1 = B1, functionB1r = functionB1r)
ggplot(data = df, mapping = aes(x = B1, y = functionB1r)) +
  geom_line() +
  geom_line(mapping = aes(x = y1/(lambda+1)))
```



note: $\sum_{j=1}^{p}(y_i - \beta_j)^2 + \lambda \sum_{j=1}^{p} \beta_j^2$ represented by line, and line represents $\beta_1^r = y_1/(1+\lambda)$. B_1 is at min of function

(b) plot 6.13 as a function of β_1

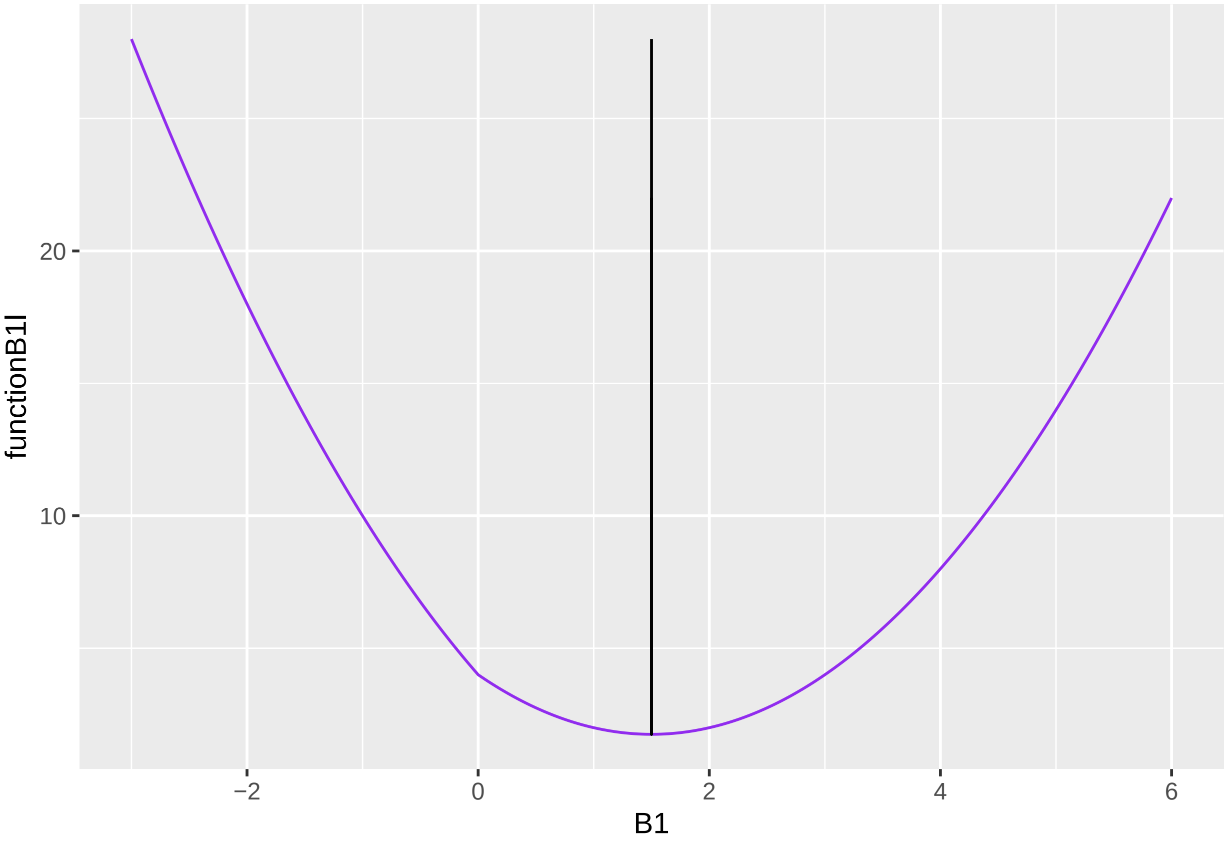
```
B1 <- seq(from = -3, to = 6, by = .01)
functionB1l <- (y1)^2 -2*y1*B1 + B1^2 + abs(B1)
bayarea <- (y1 - B1)^2 + lambda*abs(B1)
min(bayarea)
```

```
## [1] 1.75
```

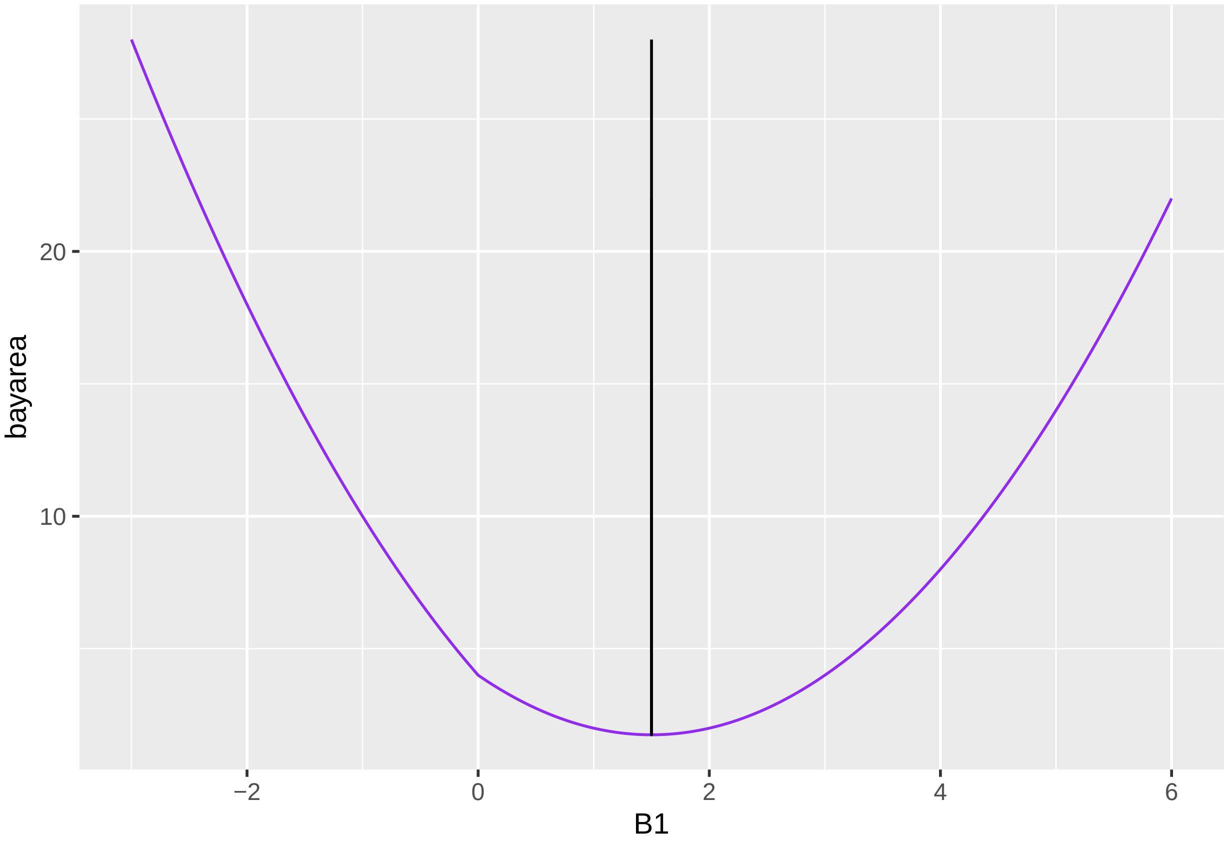
```
min(functionB1l)
```

```
## [1] 1.75
```

```
dfl <- data.frame(B1 = B1, functionB1l = functionB1l)
ggplot(data = dfl, mapping = aes(x = B1, y = functionB1l)) +
  geom_line(col = "purple2") +
  geom_line(mapping = aes(x = (y1 - (lambda/2))))
```



```
dfltest <- data.frame(B1 = B1, bayarea = bayarea)
ggplot(data = dfltest, mapping = aes(x = B1, y = bayarea)) +
  geom_line(col = "purple2") +
  geom_line(mapping = aes( x = (y1 - (lambda/2))))
```



Additional Exercise

Using the glmnet package, construct a ridge regression and LASSO model to predict violent crime in the training data set. A description for how to use this package can be found in your book on pages 251 - 255.

```
How many variables were selected by the LASSO?
What are the training MSEs for ridge and LASSO using the optimal value of (\lambda)?
If the MSEs differed, why do you think one is higher than the other in this setting?
```

```r
library(glmnet)
```

```
## Loading required package: Matrix
```

```
##
## Attaching package: 'Matrix'
```

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

```
## Loading required package: foreach
```

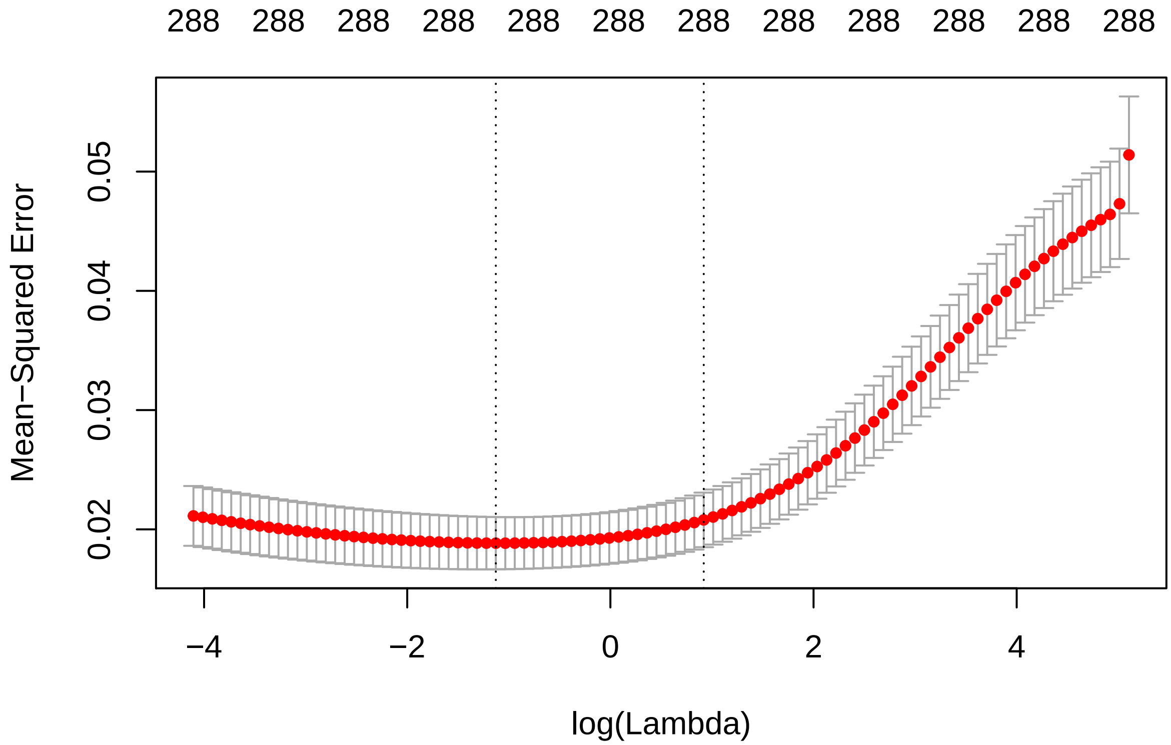
```
##
## Attaching package: 'foreach'
```

```
## The following objects are masked from 'package:purrr':
##
##     accumulate, when
## Loaded glmnet 2.0-18
```

```
grid <- 10^seq(10,-2, length =100)
xx <- model.matrix(ViolentCrimesPerPop ~., dw)[,-1]
yy = dw$ViolentCrimesPerPop
ridge.mod = glmnet(x = xx, y = yy, alpha = 0,
                   lambda = grid)
dim(coef(ridge.mod))
```

```
## [1] 289 100
```

```
set.seed(11)
train <- sample(1:nrow(xx), nrow(xx)/2)
test <- (-train)
yy.test = yy[test]
cvoutr <- cv.glmnet(xx[train,], yy[train], alpha =0)
plot(cvoutr)
```



```
bestlamr <- cvoutr$lambda.min
bestlamr
```

```
## [1] 0.3237577
```

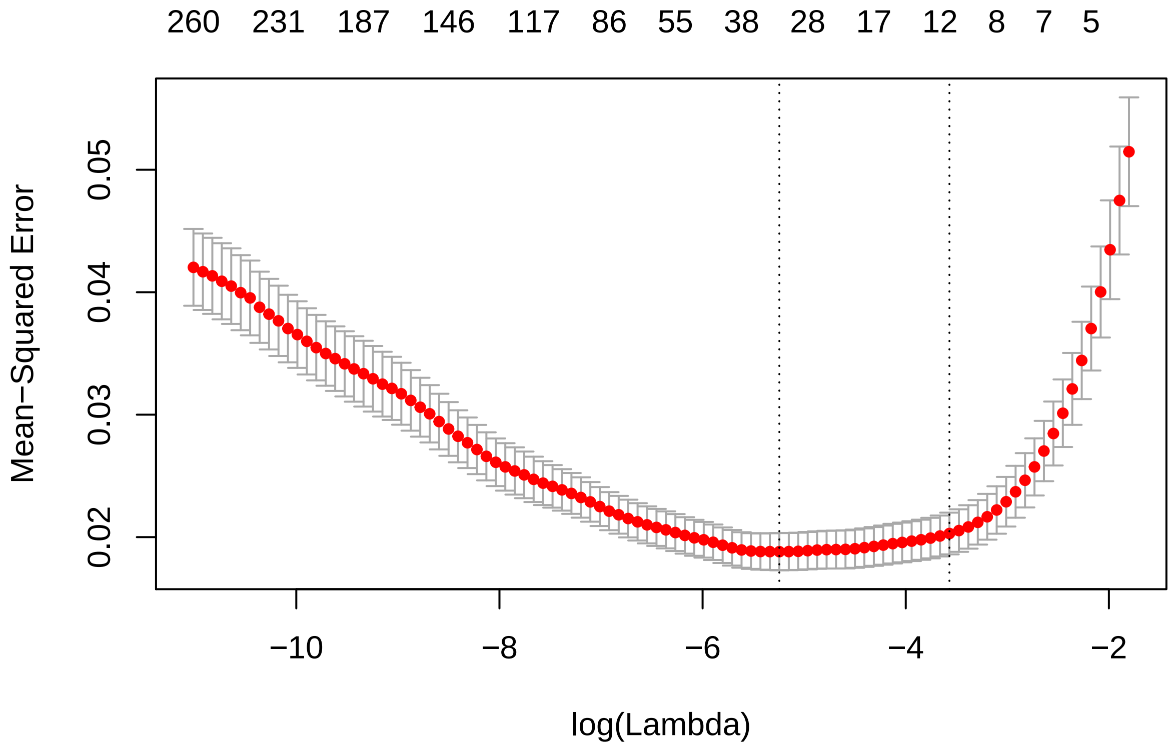
```
ridgepred = predict(ridge.mod, s = bestlamr, newx = xx[test,])
testmseridge <- mean((ridgepred - yy.test)^2)
testmseridge
```

```
## [1] 0.01849113
```

```
#now for lasso
lasso.mod = glmnet(x = xx, y = yy, alpha = 1,
                   lambda = grid)
dim(coef(lasso.mod))
```

```
## [1] 289 100
```

```
set.seed(11)
cvoutlasso <- cv.glmnet(xx[train,], yy[train], alpha =1)
plot(cvoutlasso)
```



```
bestlamlasso <- cvoutlasso$lambda.min
bestlamlasso
```

```
## [1] 0.005276443
```

```
lassopred = predict(lasso.mod, s = bestlamr, newx = xx[test,])
testmselasso <- mean((lassopred - yy.test)^2)
testmselasso
```

```
## [1] 0.05865972
```

```
lassocoeff = predict(lasso.mod, type = "coefficients", s = bestlamlasso)
lassocoeff[lassocoeff != 0]
```

```
## <sparse>[ <logic> ] : .M.sub.i.logical() maybe inefficient

##  [1]  0.3691481182 -0.0570796828 -0.0007791917  0.0326701084  0.1450908366
##  [6] -0.0077677206  0.1808884867  0.1246412015 -0.0159766045  0.0419122868
## [11]  0.0050247054 -0.0602653933 -0.3124524337  0.0025875656  0.0764096092
## [16]  0.0420028221  0.0028152885  0.1453804569
```

18 variables for lasso model. mse for ridge:.01849; mse for lasso: .0586 My guess for why the test MSE's differed was because the best lambda for lasso ended up being super small when compared to the lambda for ridge. This means that my lasso model ended up being more complex than the ridge model. Also, the lasso model should have eliminated any needless predictors, so since it was more complex, tried modeling the data more closely compared to the ridge model. However, it may have still been overfitting the test data, which could explain the higher test mse.

```r
group_F_automated_fit <- function(training_data){
  mb <- lm(ViolentCrimesPerPop ~ population +
             racePctWhite +
             agePct12t29 +
             numbUrban +
             medIncome +
             PctEmploy +
             pctWSocSec +
             medFamInc +
             whitePerCap +
             PctEmploy +
             MalePctDivorce +
             MalePctNevMarr +
             PctWorkMom +
             PctIlleg +
             PctImmigRec5 +
             PctImmigRec8 +
             PersPerOccupHous +
             PctHousLess3BR +
             PctVacantBoarded +
             PctHousOccup +
             PctVacMore6Mos +
             RentLowQ +
             MedRent +
             MedOwnCostPctIncNoMtg +
             NumStreet,
           training_data)
  mb
}
```

```r
group_F_process <- function(data){
  data %>%
    mutate(T_MPD = MalePctDivorce^(1/3),
           T_PK2P = PctKids2Par^(1/3),
           T_PPDH = PctPersDenseHous^2)
}


group_F_fit2.0 <- function(training_data) {
  m1 <- lm(ViolentCrimesPerPop ~  T_MPD + T_PK2P +
             PctIlleg + T_PPDH + RentLowQ+MedRent, training_data)
  m1
}
```

```r
group_F_fit <- function(training_data) {
  m1 <- lm(ViolentCrimesPerPop~ I(MalePctDivorce^(1/3)) + I(PctKids2Par^(1/3)) +
             PctIlleg + (PctPersDenseHous^2) + RentLowQ+MedRent, training_data)
  m1
}
```

```r
group_F_MSE <- function(model, data) {
  MSE<-mean((data$ViolentCrimesPerPop - predict.lm(model, data)) ^ 2)
  return (MSE)
}
```

```r
library(leaps)
#forward
regsubsets(ViolentCrimesPerPop ~ population +
             householdsize +
             racepctblack +
             racePctWhite +
             racePctAsian +
             racePctHisp +
             agePct12t21 +
             agePct12t29 +
             agePct16t24 +
             agePct65up +
             numbUrban +
             pctUrban +
             medIncome +
             pctWWage +
             pctWFarmSelf +
             pctWInvInc +
             pctWSocSec +
             pctWPubAsst +
             pctWRetire +
             medFamInc +
             perCapInc +
             whitePerCap +
             blackPerCap +
             indianPerCap +
             AsianPerCap +
             OtherPerCap +
             HispPerCap +
             NumUnderPov +
             PctPopUnderPov +
             PctLess9thGrade +
             PctNotHSGrad +
             PctBSorMore +
             PctUnemployed +
             PctEmploy +
             PctEmplManu +
             PctEmplProfServ +
             PctOccupManu +
             PctOccupMgmtProf +
             MalePctDivorce +
             MalePctNevMarr +
             FemalePctDiv +
             TotalPctDiv +
             PersPerFam +
             PctFam2Par +
             PctKids2Par +
             PctYoungKids2Par +
             PctTeen2Par +
             PctWorkMomYoungKids +
             PctWorkMom +
             NumIlleg +
             PctIlleg +
             NumImmig +
             PctImmigRecent +
             PctImmigRec5 +
             PctImmigRec8 +
             PctImmigRec10 +
             PctRecentImmig +
             PctRecImmig5 +
             PctRecImmig8 +
             PctRecImmig10 +
             PctSpeakEnglOnly +
             PctNotSpeakEnglWell +
             PctLargHouseFam +
             PctLargHouseOccup +
             PersPerOccupHous +
             PersPerOwnOccHous +
             PersPerRentOccHous +
             PctPersOwnOccup +
             PctPersDenseHous +
             PctHousLess3BR +
             MedNumBR +
             HousVacant +
             PctHousOccup +
             PctHousOwnOcc +
             PctVacantBoarded +
             PctVacMore6Mos +
             MedYrHousBuilt +
             PctHousNoPhone +
             PctWOFullPlumb +
             OwnOccLowQuart +
             OwnOccMedVal +
             OwnOccHiQuart +
             RentLowQ +
             RentMedian +
             RentHighQ +
             MedRent +
             MedRentPctHousInc +
             MedOwnCostPctInc +
             MedOwnCostPctIncNoMtg +
             NumInShelters +
             NumStreet +
             PctForeignBorn +
             PctBornSameState +
             PctSameHouse85 +
             PctSameCity85 +
             PctSameState85,
           data = d, nvmax = 25, method = "forward")
```

```
## Subset selection object
## Call: regsubsets.formula(ViolentCrimesPerPop ~ population + householdsize +
##     racepctblack + racePctWhite + racePctAsian + racePctHisp +
##     agePct12t21 + agePct12t29 + agePct16t24 + agePct65up + numbUrban +
##     pctUrban + medIncome + pctWWage + pctWFarmSelf + pctWInvInc +
##     pctWSocSec + pctWPubAsst + pctWRetire + medFamInc + perCapInc +
##     whitePerCap + blackPerCap + indianPerCap + AsianPerCap +
##     OtherPerCap + HispPerCap + NumUnderPov + PctPopUnderPov +
##     PctLess9thGrade + PctNotHSGrad + PctBSorMore + PctUnemployed +
##     PctEmploy + PctEmplManu + PctEmplProfServ + PctOccupManu +
##     PctOccupMgmtProf + MalePctDivorce + MalePctNevMarr + FemalePctDiv +
##     TotalPctDiv + PersPerFam + PctFam2Par + PctKids2Par + PctYoungKids2Par +
##     PctTeen2Par + PctWorkMomYoungKids + PctWorkMom + NumIlleg +
##     PctIlleg + NumImmig + PctImmigRecent + PctImmigRec5 + PctImmigRec8 +
##     PctImmigRec10 + PctRecentImmig + PctRecImmig5 + PctRecImmig8 +
##     PctRecImmig10 + PctSpeakEnglOnly + PctNotSpeakEnglWell +
##     PctLargHouseFam + PctLargHouseOccup + PersPerOccupHous +
##     PersPerOwnOccHous + PersPerRentOccHous + PctPersOwnOccup +
##     PctPersDenseHous + PctHousLess3BR + MedNumBR + HousVacant +
##     PctHousOccup + PctHousOwnOcc + PctVacantBoarded + PctVacMore6Mos +
##     MedYrHousBuilt + PctHousNoPhone + PctWOFullPlumb + OwnOccLowQuart +
##     OwnOccMedVal + OwnOccHiQuart + RentLowQ + RentMedian + RentHighQ +
##     MedRent + MedRentPctHousInc + MedOwnCostPctInc + MedOwnCostPctIncNoMtg +
##     NumInShelters + NumStreet + PctForeignBorn + PctBornSameState +
##     PctSameHouse85 + PctSameCity85 + PctSameState85, data = d,
##     nvmax = 25, method = "forward")
## 96 Variables  (and intercept)
##                       Forced in Forced out
## population                FALSE      FALSE
## householdsize             FALSE      FALSE
## racepctblack              FALSE      FALSE
## racePctWhite              FALSE      FALSE
## racePctAsian              FALSE      FALSE
## racePctHisp               FALSE      FALSE
## agePct12t21               FALSE      FALSE
## agePct12t29               FALSE      FALSE
## agePct16t24               FALSE      FALSE
## agePct65up                FALSE      FALSE
## numbUrban                 FALSE      FALSE
## pctUrban                  FALSE      FALSE
## medIncome                 FALSE      FALSE
## pctWWage                  FALSE      FALSE
## pctWFarmSelf              FALSE      FALSE
## pctWInvInc                FALSE      FALSE
## pctWSocSec                FALSE      FALSE
## pctWPubAsst               FALSE      FALSE
## pctWRetire                FALSE      FALSE
## medFamInc                 FALSE      FALSE
## perCapInc                 FALSE      FALSE
## whitePerCap               FALSE      FALSE
## blackPerCap               FALSE      FALSE
## indianPerCap              FALSE      FALSE
## AsianPerCap               FALSE      FALSE
## OtherPerCap               FALSE      FALSE
## HispPerCap                FALSE      FALSE
## NumUnderPov               FALSE      FALSE
## PctPopUnderPov            FALSE      FALSE
## PctLess9thGrade           FALSE      FALSE
## PctNotHSGrad              FALSE      FALSE
## PctBSorMore               FALSE      FALSE
## PctUnemployed             FALSE      FALSE
## PctEmploy                 FALSE      FALSE
## PctEmplManu               FALSE      FALSE
## PctEmplProfServ           FALSE      FALSE
## PctOccupManu              FALSE      FALSE
## PctOccupMgmtProf          FALSE      FALSE
## MalePctDivorce            FALSE      FALSE
## MalePctNevMarr            FALSE      FALSE
## FemalePctDiv              FALSE      FALSE
## TotalPctDiv               FALSE      FALSE
## PersPerFam                FALSE      FALSE
## PctFam2Par                FALSE      FALSE
## PctKids2Par               FALSE      FALSE
## PctYoungKids2Par          FALSE      FALSE
## PctTeen2Par               FALSE      FALSE
## PctWorkMomYoungKids       FALSE      FALSE
## PctWorkMom                FALSE      FALSE
## NumIlleg                  FALSE      FALSE
## PctIlleg                  FALSE      FALSE
## NumImmig                  FALSE      FALSE
## PctImmigRecent            FALSE      FALSE
## PctImmigRec5              FALSE      FALSE
## PctImmigRec8              FALSE      FALSE
## PctImmigRec10             FALSE      FALSE
## PctRecentImmig            FALSE      FALSE
## PctRecImmig5              FALSE      FALSE
## PctRecImmig8              FALSE      FALSE
## PctRecImmig10             FALSE      FALSE
## PctSpeakEnglOnly          FALSE      FALSE
## PctNotSpeakEnglWell       FALSE      FALSE
## PctLargHouseFam           FALSE      FALSE
## PctLargHouseOccup         FALSE      FALSE
## PersPerOccupHous          FALSE      FALSE
## PersPerOwnOccHous         FALSE      FALSE
## PersPerRentOccHous        FALSE      FALSE
## PctPersOwnOccup           FALSE      FALSE
## PctPersDenseHous          FALSE      FALSE
## PctHousLess3BR            FALSE      FALSE
## MedNumBR                  FALSE      FALSE
## HousVacant                FALSE      FALSE
## PctHousOccup              FALSE      FALSE
## PctHousOwnOcc             FALSE      FALSE
## PctVacantBoarded          FALSE      FALSE
## PctVacMore6Mos            FALSE      FALSE
## MedYrHousBuilt            FALSE      FALSE
## PctHousNoPhone            FALSE      FALSE
## PctWOFullPlumb            FALSE      FALSE
## OwnOccLowQuart            FALSE      FALSE
## OwnOccMedVal              FALSE      FALSE
## OwnOccHiQuart             FALSE      FALSE
## RentLowQ                  FALSE      FALSE
## RentMedian                FALSE      FALSE
## RentHighQ                 FALSE      FALSE
## MedRent                   FALSE      FALSE
## MedRentPctHousInc         FALSE      FALSE
## MedOwnCostPctInc          FALSE      FALSE
## MedOwnCostPctIncNoMtg     FALSE      FALSE
## NumInShelters             FALSE      FALSE
## NumStreet                 FALSE      FALSE
## PctForeignBorn            FALSE      FALSE
## PctBornSameState          FALSE      FALSE
## PctSameHouse85            FALSE      FALSE
## PctSameCity85             FALSE      FALSE
## PctSameState85            FALSE      FALSE
## 1 subsets of each size up to 25
## Selection Algorithm: forward
```

```
forwardmodelfit <- function(training_data){
  mf <- lm(ViolentCrimesPerPop ~ NumStreet +
             MedOwnCostPctIncNoMtg +
             MedRentPctHousInc +
             MedRent +
             RentLowQ +
             PctHousNoPhone +
             PctVacMore6Mos +
             PctVacantBoarded +
             PctHousOccup +
             PctImmigRec10 +
             PctImmigRec8 +
             PctImmigRec5 +
             PctIlleg +
             PctWorkMom +
             FemalePctDiv +
             MalePctDivorce +
             PctEmplProfServ +
             PctEmploy +
             PctNotHSGrad +
             PctLess9thGrade +
             indianPerCap +
             pctWSocSec +
             pctUrban +
             racePctHisp +
             racePctWhite,
           training_data)
  mf
}
group_F_MSE(forwardmodelfit(d), d)
```

```
## [1] 0.01688268
```

```
backwardmodelfit <- function(training_data){
  mb <- lm(ViolentCrimesPerPop ~ population +
             racePctWhite +
             agePct12t29 +
             numbUrban +
             medIncome +
             PctEmploy +
             pctWSocSec +
             medFamInc +
             whitePerCap +
```

```r
             PctEmploy +
             MalePctDivorce +
             MalePctNevMarr +
             PctWorkMom +
             PctIlleg +
             PctImmigRec5 +
             PctImmigRec8 +
             PersPerOccupHous +
             PctHousLess3BR +
             PctVacantBoarded +
             PctHousOccup +
             PctVacMore6Mos +
             RentLowQ +
             MedRent +
             MedOwnCostPctIncNoMtg +
             NumStreet,
           training_data)
  mb
}
group_F_MSE(backwardmodelfit(d), d)
```

```
## [1] 0.01684645
```

```r
group_F_MSE(group_F_fit2.0(group_F_process(d)), group_F_process(d))
```

```
## [1] 0.02039485
```

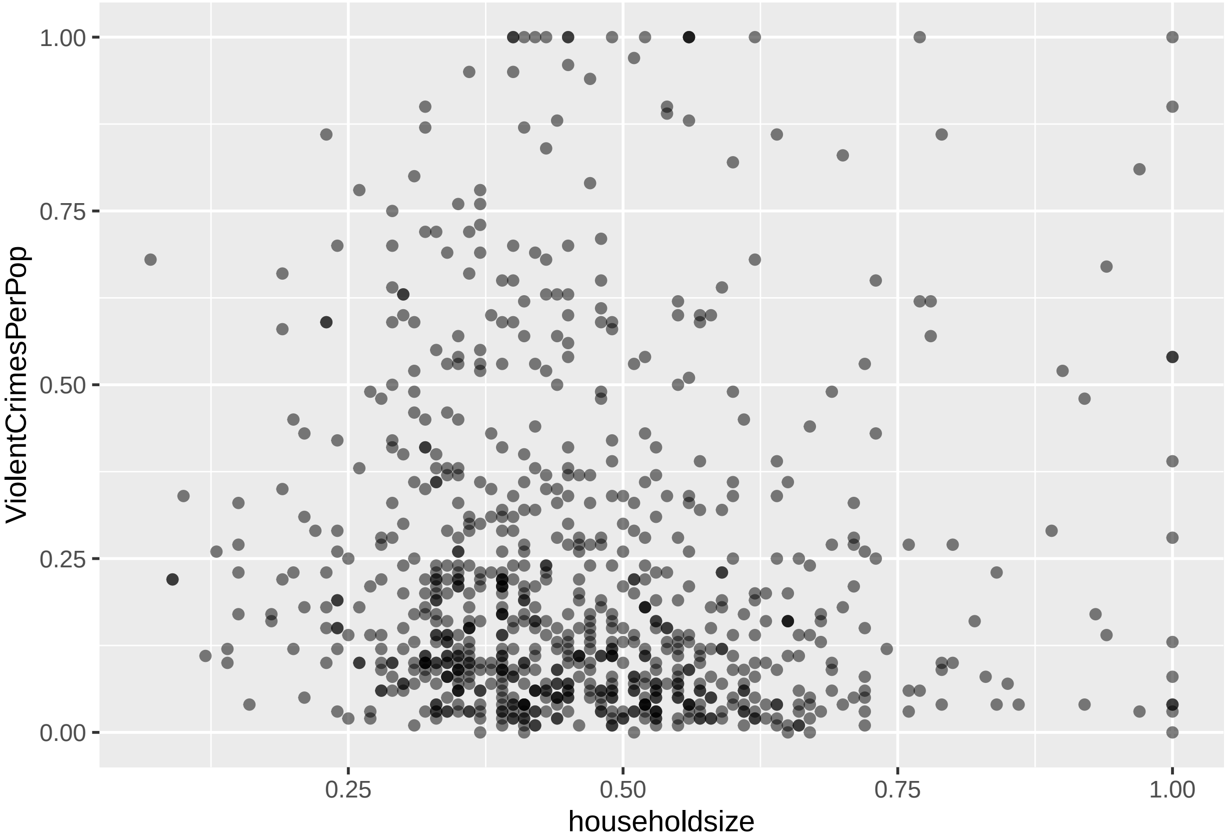
```r
summary(forwardmodelfit(d))
```

```
## 
## Call:
## lm(formula = ViolentCrimesPerPop ~ NumStreet + MedOwnCostPctIncNoMtg + 
##     MedRentPctHousInc + MedRent + RentLowQ + PctHousNoPhone + 
##     PctVacMore6Mos + PctVacantBoarded + PctHousOccup + PctImmigRec10 + 
##     PctImmigRec8 + PctImmigRec5 + PctIlleg + PctWorkMom + FemalePctDiv + 
##     MalePctDivorce + PctEmplProfServ + PctEmploy + PctNotHSGrad + 
##     PctLess9thGrade + indianPerCap + pctWSocSec + pctUrban + 
##     racePctHisp + racePctWhite, data = training_data)
## 
## Residuals:
##      Min       1Q   Median       3Q      Max 
## -0.50745 -0.07539 -0.01158  0.05254  0.63131 
## 
## Coefficients:
##                        Estimate Std. Error t value Pr(>|t|)    
## (Intercept)            0.007454   0.093093   0.080 0.936206    
## NumStreet              0.190690   0.040275   4.735 2.61e-06 ***
## MedOwnCostPctIncNoMtg -0.109093   0.030841  -3.537 0.000428 ***
## MedRentPctHousInc      0.075303   0.038247   1.969 0.049325 *  
## MedRent                0.435922   0.090754   4.803 1.87e-06 ***
## RentLowQ              -0.461313   0.090337  -5.107 4.13e-07 ***
## PctHousNoPhone        -0.056889   0.043824  -1.298 0.194628    
## PctVacMore6Mos        -0.060606   0.035142  -1.725 0.084998 .  
## PctVacantBoarded       0.066917   0.030083   2.224 0.026408 *  
## PctHousOccup          -0.105796   0.032763  -3.229 0.001294 ** 
## PctImmigRec10         -0.112990   0.076586  -1.475 0.140532
## PctImmigRec8           0.386945   0.110547   3.500 0.000491 ***
## PctImmigRec5          -0.233562   0.074136  -3.150 0.001693 **
## PctIlleg               0.324441   0.046463   6.983 6.24e-12 ***
## PctWorkMom            -0.102374   0.039767  -2.574 0.010227 *
## FemalePctDiv          -0.018782   0.085610  -0.219 0.826409
## MalePctDivorce         0.294348   0.083708   3.516 0.000463 ***
## PctEmplProfServ        0.068993   0.041896   1.647 0.100010
## PctEmploy              0.208276   0.082885   2.513 0.012179 *
## PctNotHSGrad           0.327993   0.110782   2.961 0.003163 **
## PctLess9thGrade       -0.156711   0.084835  -1.847 0.065094 .
## indianPerCap          -0.062586   0.032145  -1.947 0.051900 .
## pctWSocSec             0.169979   0.062461   2.721 0.006647 **
## pctUrban               0.041869   0.013461   3.110 0.001938 **
## racePctHisp            0.071670   0.034060   2.104 0.035684 *
## racePctWhite          -0.217587   0.040351  -5.392 9.25e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1321 on 774 degrees of freedom
## Multiple R-squared:  0.6937, Adjusted R-squared:  0.6838
## F-statistic: 70.13 on 25 and 774 DF,  p-value: < 2.2e-16
```

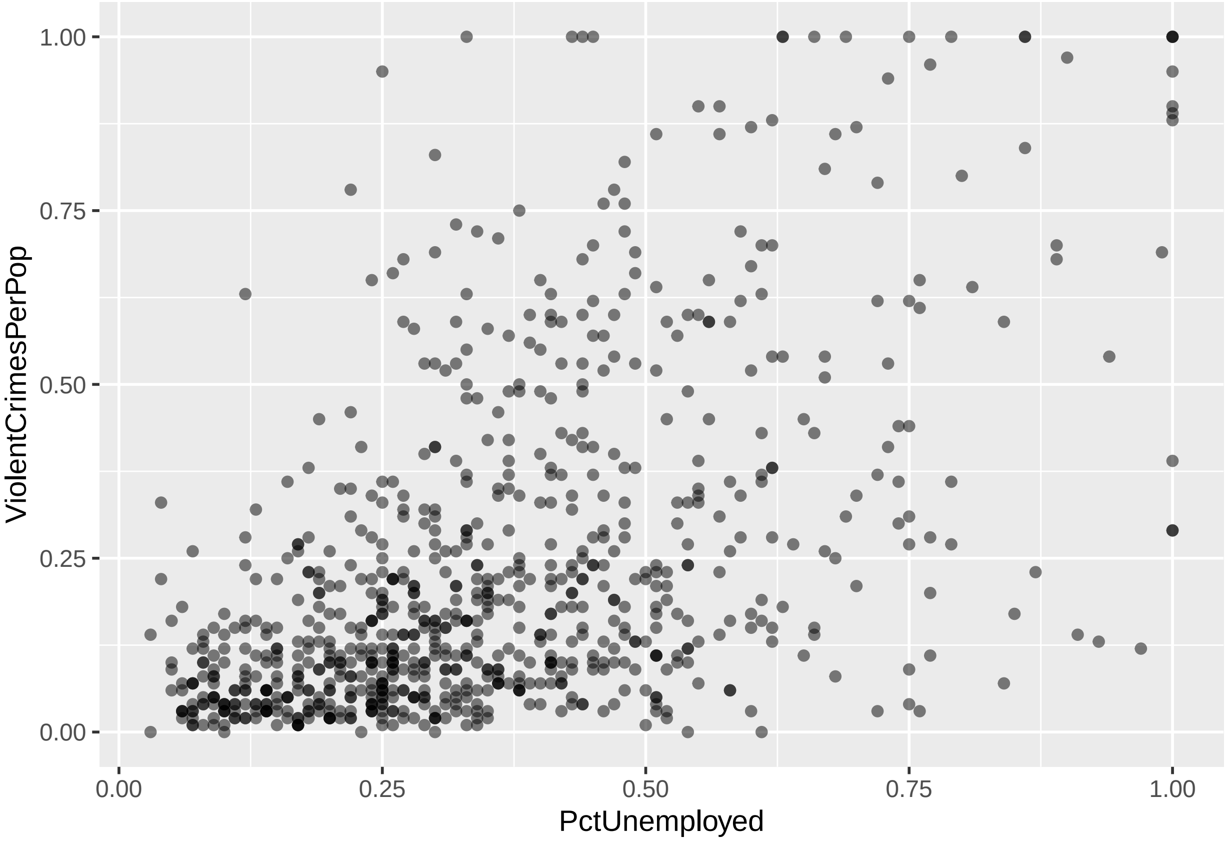
```
summary(backwardmodelfit(d))
```

```
##
## Call:
## lm(formula = ViolentCrimesPerPop ~ population + racePctWhite +
##     agePct12t29 + numbUrban + medIncome + PctEmploy + pctWSocSec +
##     medFamInc + whitePerCap + PctEmploy + MalePctDivorce + MalePctNevMarr +
##     PctWorkMom + PctIlleg + PctImmigRec5 + PctImmigRec8 + PersPerOccupHous +
##     PctHousLess3BR + PctVacantBoarded + PctHousOccup + PctVacMore6Mos +
##     RentLowQ + MedRent + MedOwnCostPctIncNoMtg + NumStreet, data = training_data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.48416 -0.07536 -0.01504  0.05005  0.64174
##
## Coefficients:
##                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)            0.11404    0.12326   0.925 0.355128
## population            -1.25891    0.35833  -3.513 0.000468 ***
## racePctWhite          -0.24031    0.04495  -5.347 1.18e-07 ***
## agePct12t29           -0.19332    0.09133  -2.117 0.034601 *
## numbUrban              1.29995    0.35654   3.646 0.000284 ***
## medIncome             -0.41544    0.21448  -1.937 0.053113 .
## PctEmploy              0.15422    0.07800   1.977 0.048385 *
## pctWSocSec             0.19251    0.07288   2.642 0.008420 **
## medFamInc              0.56811    0.20508   2.770 0.005737 **
## whitePerCap           -0.24261    0.09689  -2.504 0.012488 *
## MalePctDivorce         0.29898    0.04932   6.062 2.10e-09 ***
## MalePctNevMarr         0.12344    0.06899   1.789 0.073955 .
## PctWorkMom            -0.10614    0.04140  -2.564 0.010548 *
## PctIlleg               0.34401    0.04835   7.114 2.56e-12 ***
## PctImmigRec5          -0.22614    0.07289  -3.103 0.001988 **
## PctImmigRec8            0.28487    0.07858   3.625 0.000307 ***
## PersPerOccupHous         0.16902    0.06770   2.497 0.012740 *
## PctHousLess3BR           0.08395    0.05506   1.525 0.127770
## PctVacantBoarded         0.07285    0.03106   2.345 0.019279 *
## PctHousOccup            -0.12161    0.03311  -3.673 0.000256 ***
## PctVacMore6Mos          -0.07693    0.03448  -2.231 0.025964 *
## RentLowQ                -0.47068    0.09126  -5.158 3.18e-07 ***
## MedRent                  0.46991    0.10723   4.382 1.34e-05 ***
## MedOwnCostPctIncNoMtg -0.10267    0.03000  -3.422 0.000653 ***
## NumStreet                0.17518    0.04918   3.562 0.000391 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1319 on 775 degrees of freedom
## Multiple R-squared:  0.6944, Adjusted R-squared:  0.6849
## F-statistic: 73.37 on 24 and 775 DF,  p-value: < 2.2e-16
```

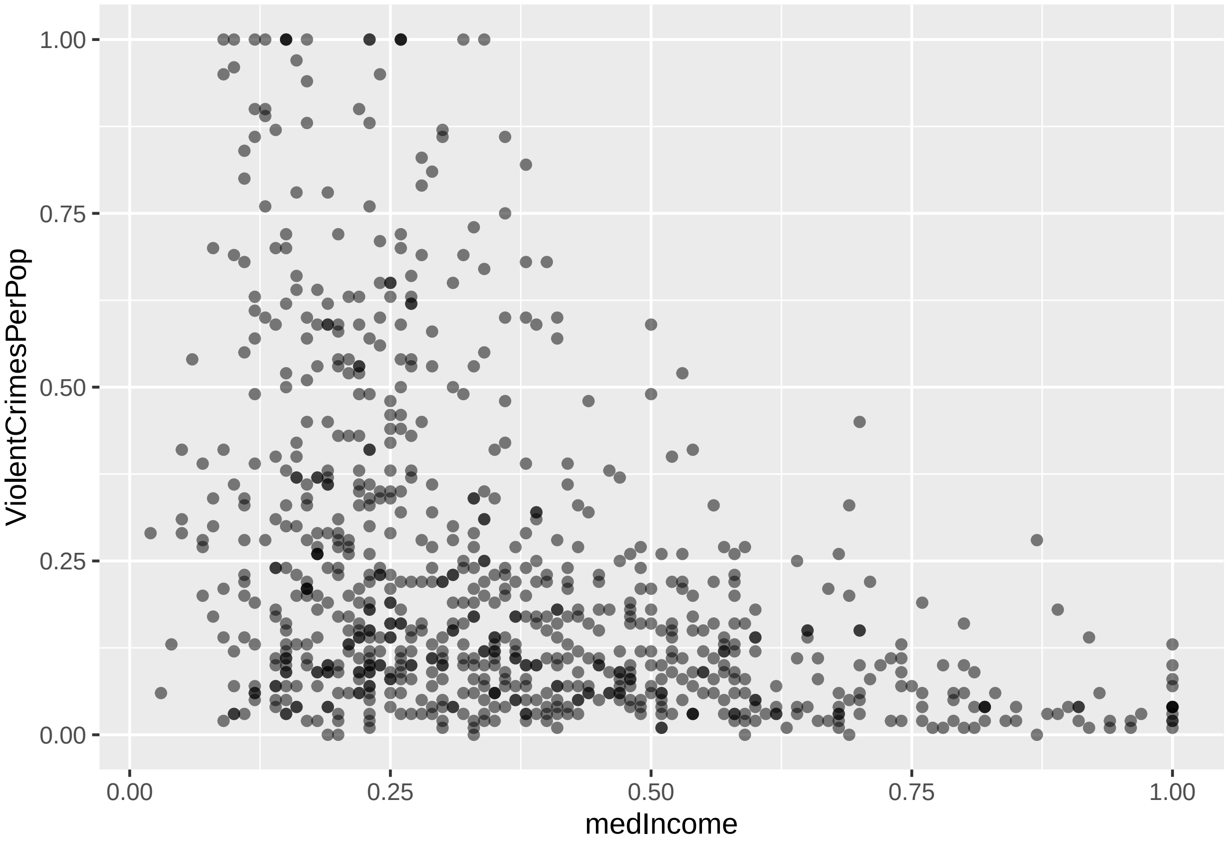
```
#Data Exploration

d <- read.csv("http://andrewpbray.github.io/data/crime-train.csv")



#maybe
ggplot(data = d, mapping = aes(x = householdsize, y = ViolentCrimesPerPop)) +
  geom_point(alpha = .5)
```



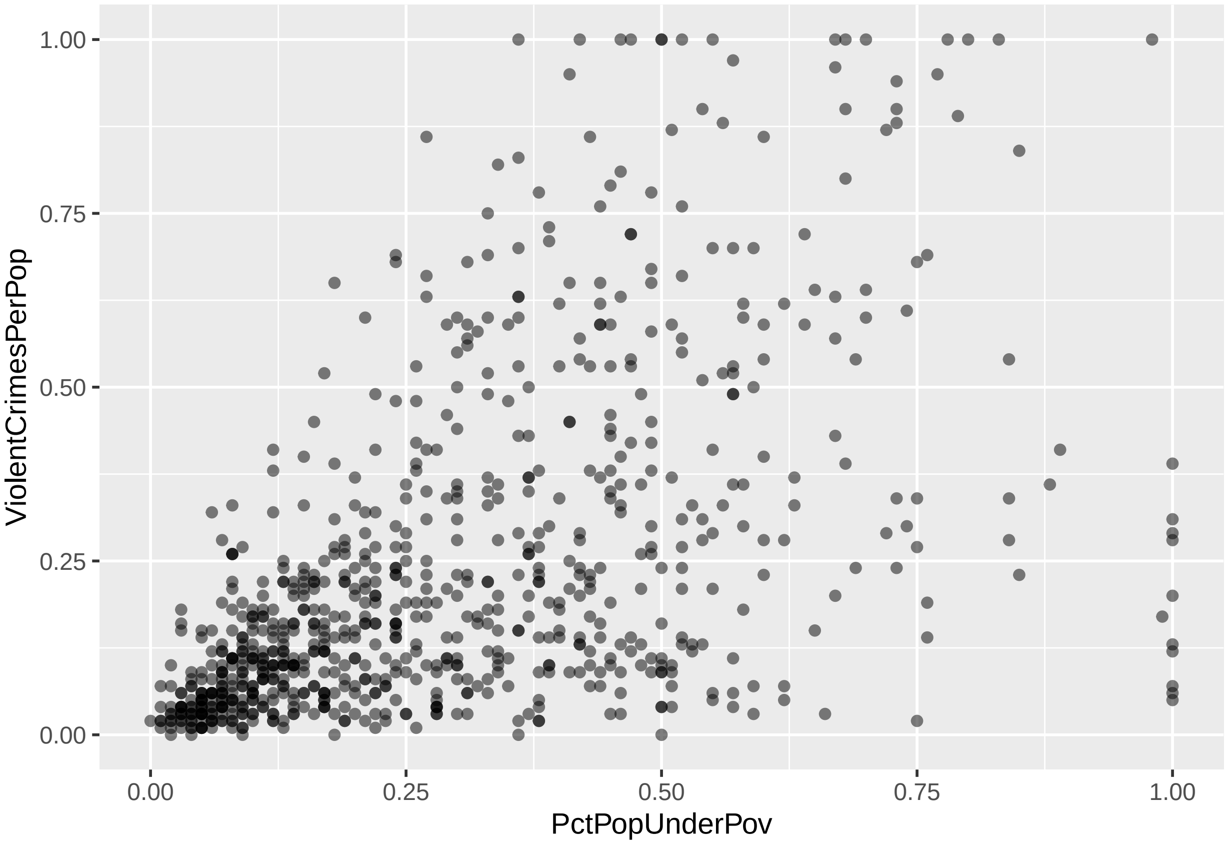
```r
#ok
ggplot(data = d, mapping = aes(x = PctUnemployed, y = ViolentCrimesPerPop)) +
  geom_point(alpha = .5)
```



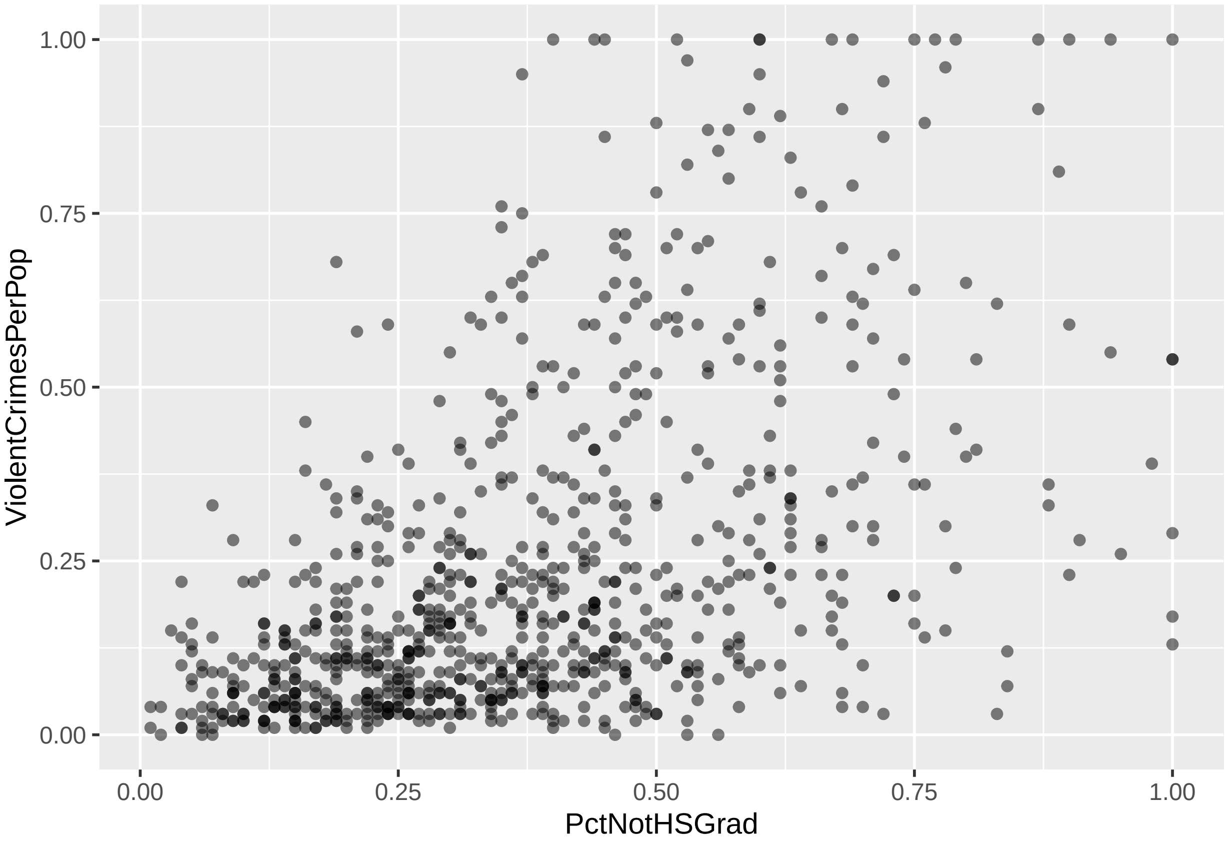
```r
#ok
ggplot(data = d, mapping = aes(x = medIncome, y = ViolentCrimesPerPop)) +
  geom_point(alpha = .5)
```



```
#ok
ggplot(data = d, mapping = aes(x = PctPopUnderPov, y = ViolentCrimesPerPop)) +
  geom_point(alpha = .5)
```

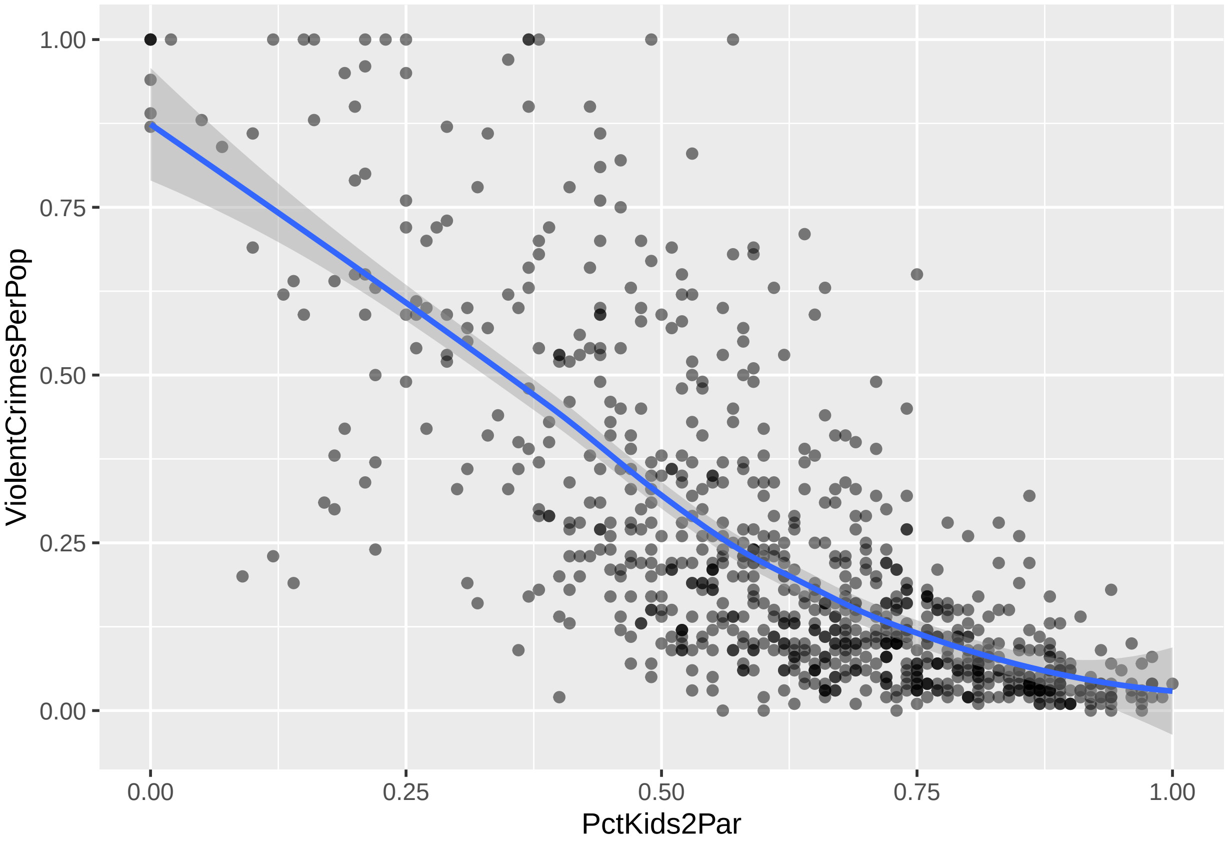


```
#ok
ggplot(data = d, mapping = aes(x = PctNotHSGrad, y = ViolentCrimesPerPop)) +
  geom_point(alpha = .5)
```

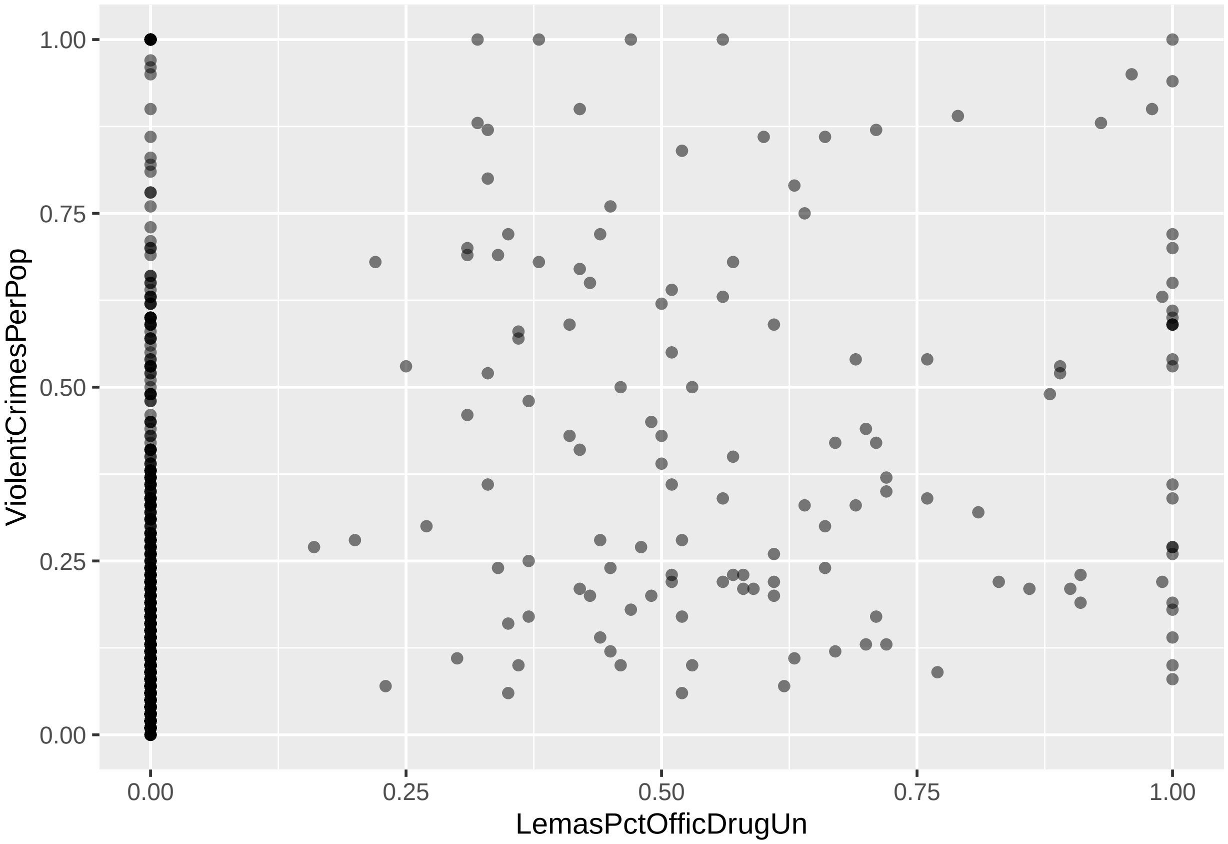


```
#ok
ggplot(data = d, mapping = aes(x = PctKids2Par, y = ViolentCrimesPerPop)) +
  geom_point(alpha = .5) +
  geom_smooth()
```

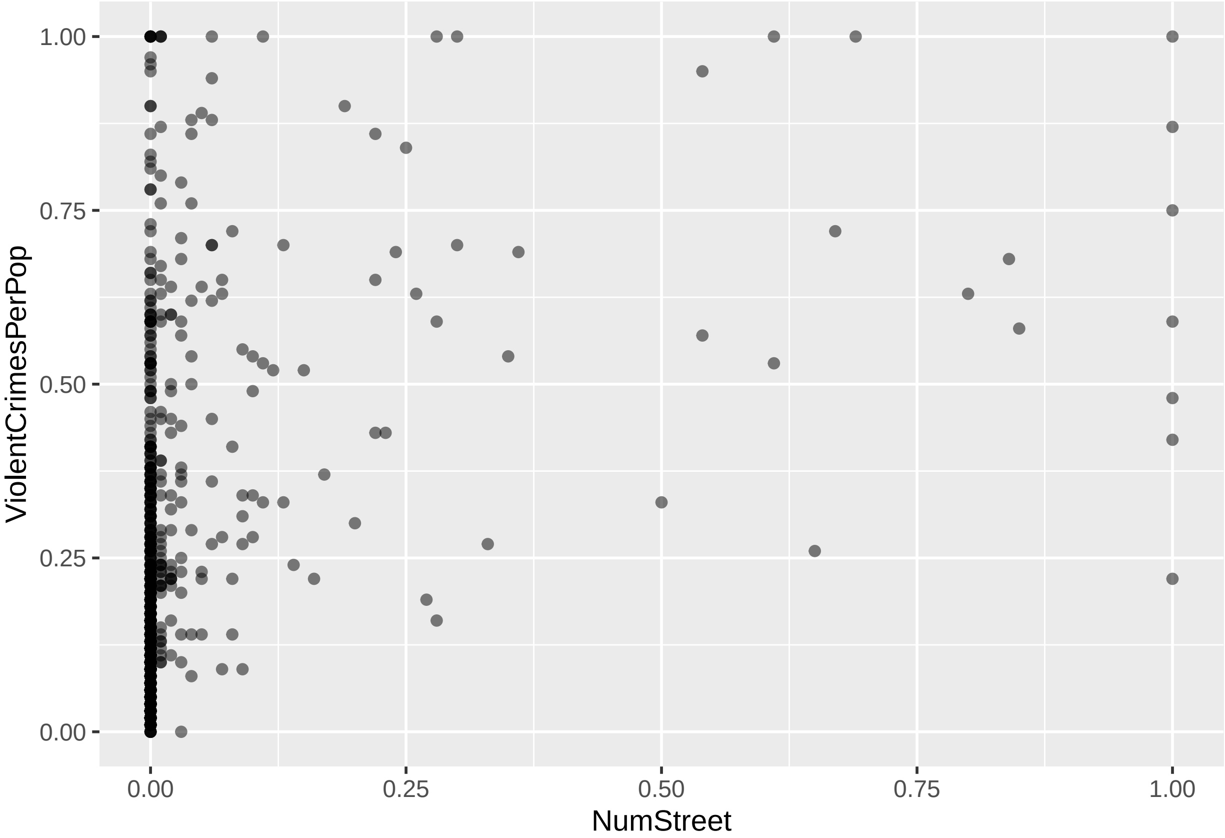
```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



```
ggplot(data = d, mapping = aes(x = LemasPctOfficDrugUn, y = ViolentCrimesPerPop)) +
  geom_point(alpha = .5)
```



```r
ggplot(data = d, mapping = aes(x = NumStreet, y = ViolentCrimesPerPop)) +
  geom_point(alpha = .5)
```



```
cor(d$PctKids2Par, d$PctNotHSGrad) #-.697
```

```
## [1] -0.6974628
```

```
cor(d$PctKids2Par, d$PctPopUnderPov) #-.771
```

```
## [1] -0.7713265
```

```
cor(d$PctKids2Par, d$medIncome) #.712
```

```
## [1] 0.7120151
```

```
cor(d$PctKids2Par, d$PctUnemployed) #-.698
```

```
## [1] -0.6979051
```

```
cor(d$ViolentCrimesPerPop, d$PctPopUnderPov) #.538
```

```
## [1] 0.5377977
```

```
cor(d$ViolentCrimesPerPop, d$medIncome) #-.414
```

```
## [1] -0.4138494
```

```
cor(d$ViolentCrimesPerPop, d$PctFam2Par) #-.705
```

```
## [1] -0.7052806
```

```
cor(d$ViolentCrimesPerPop, d$householdsize) #-0.386
```

```
## [1] -0.03861389
```

```r
cor(d$ViolentCrimesPerPop, d$PctUnemployed) #.537
```

```
## [1] 0.5373199
```

```r
cor(d$ViolentCrimesPerPop, d$PctNotHSGrad) #.5157
```

```
## [1] 0.5156772
```

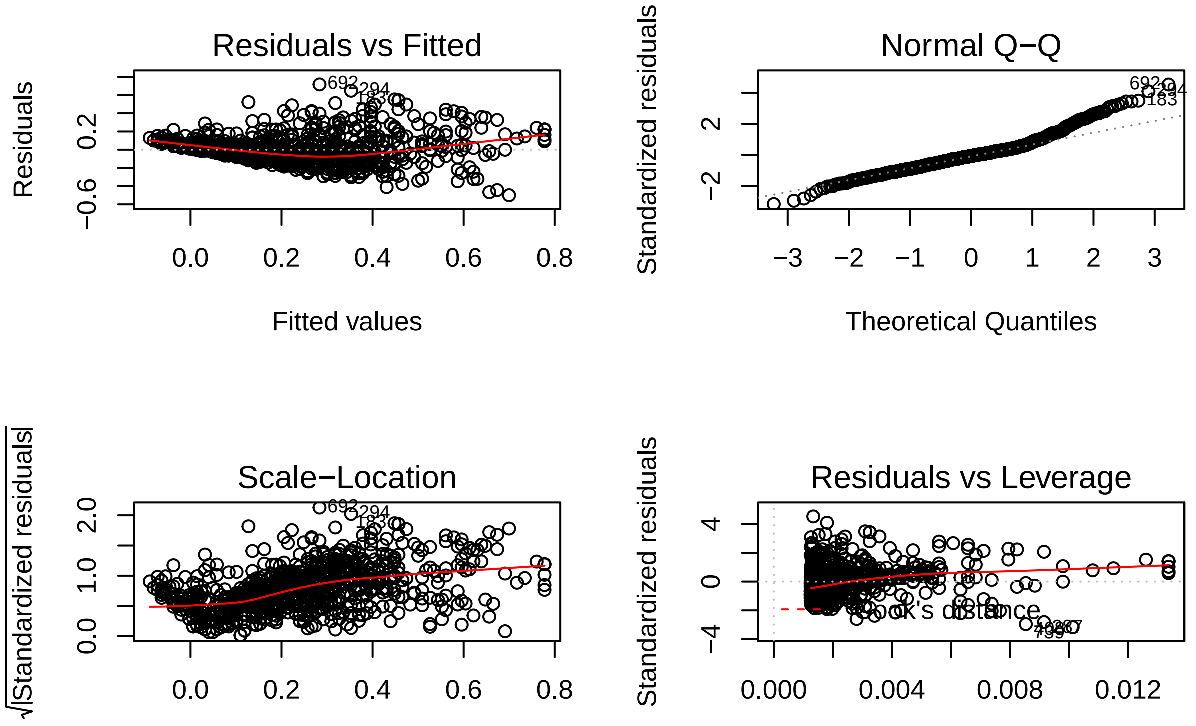
```r
cor(d$PctUnemployed, d$PctPopUnderPov) #.7839
```

```
## [1] 0.7839299
```

```r
#r2
modelkd <- lm(ViolentCrimesPerPop ~ PctKids2Par + PctPopUnderPov, d)
summary(modelkd)
```

```
##
## Call:
## lm(formula = ViolentCrimesPerPop ~ PctKids2Par + PctPopUnderPov,
##     data = d)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.47643 -0.10037 -0.01265  0.06511  0.73254
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)     0.84855    0.03776  22.472   <2e-16 ***
## PctKids2Par    -0.93959    0.04383 -21.436   <2e-16 ***
## PctPopUnderPov -0.08756    0.04086  -2.143   0.0324 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1579 on 797 degrees of freedom
## Multiple R-squared:  0.5492, Adjusted R-squared:  0.548
## F-statistic: 485.4 on 2 and 797 DF,  p-value: < 2.2e-16
```

```r
#r2 = .5656
model1 <- lm(ViolentCrimesPerPop ~ PctKids2Par, data = d)
par (mfrow =  c(2,2))
plot(model1)
```

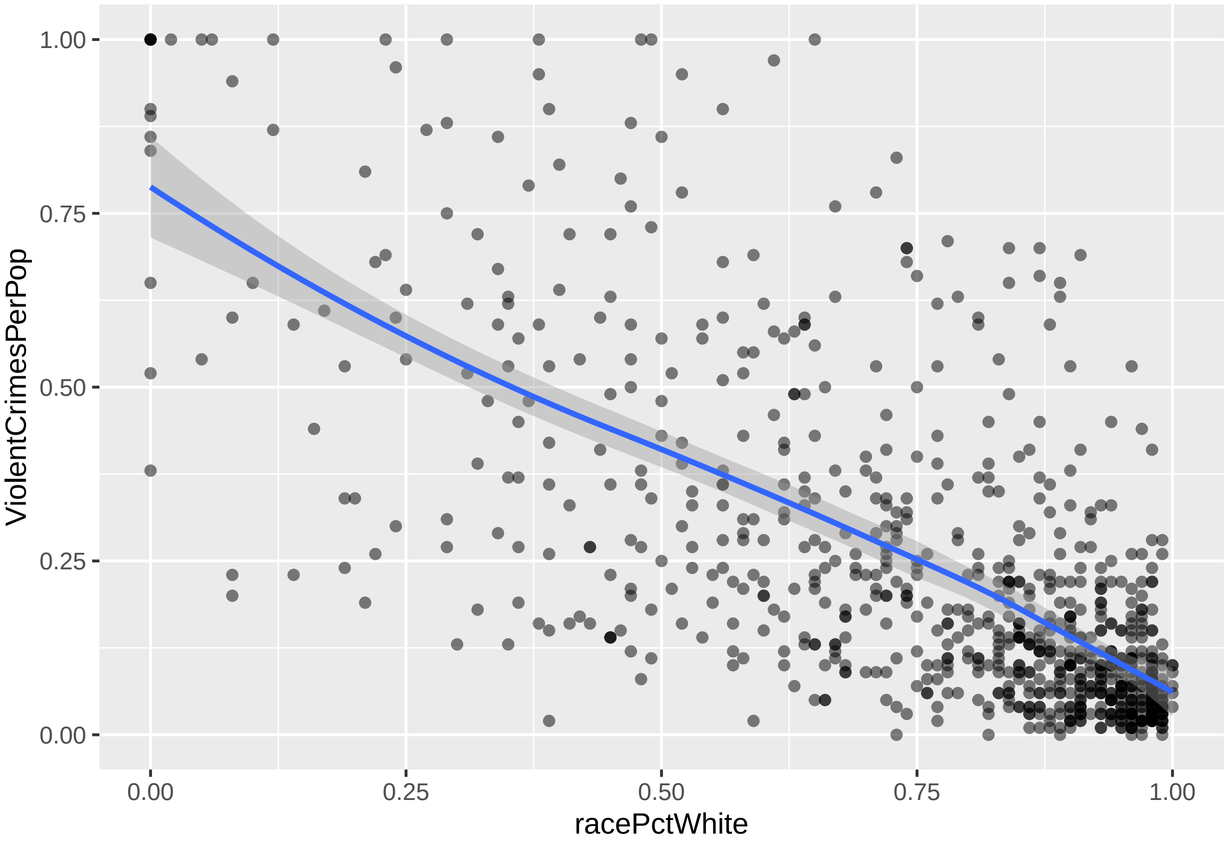


```
summary(model1)
```

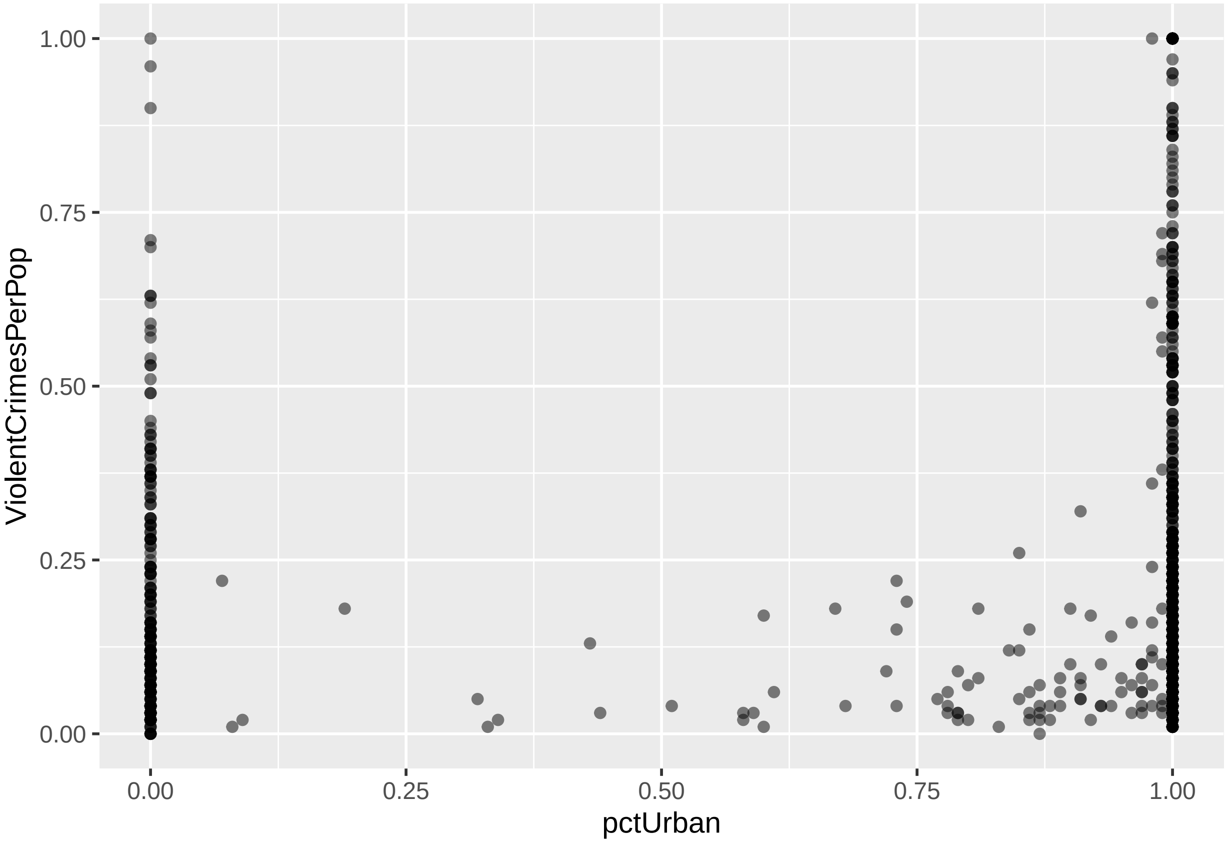
```
##
## Call:
## lm(formula = ViolentCrimesPerPop ~ PctKids2Par, data = d)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.49968 -0.09737 -0.01168  0.06528  0.71654
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.77773    0.01830   42.49   <2e-16 ***
## PctKids2Par -0.86713    0.02796  -31.01   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1583 on 798 degrees of freedom
## Multiple R-squared:  0.5466, Adjusted R-squared:  0.546
## F-statistic: 961.9 on 1 and 798 DF,  p-value: < 2.2e-16
```

```
ggplot(data = d, mapping = aes(x = racePctWhite, y = ViolentCrimesPerPop)) +
  geom_point(alpha = .5) + geom_smooth()
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```

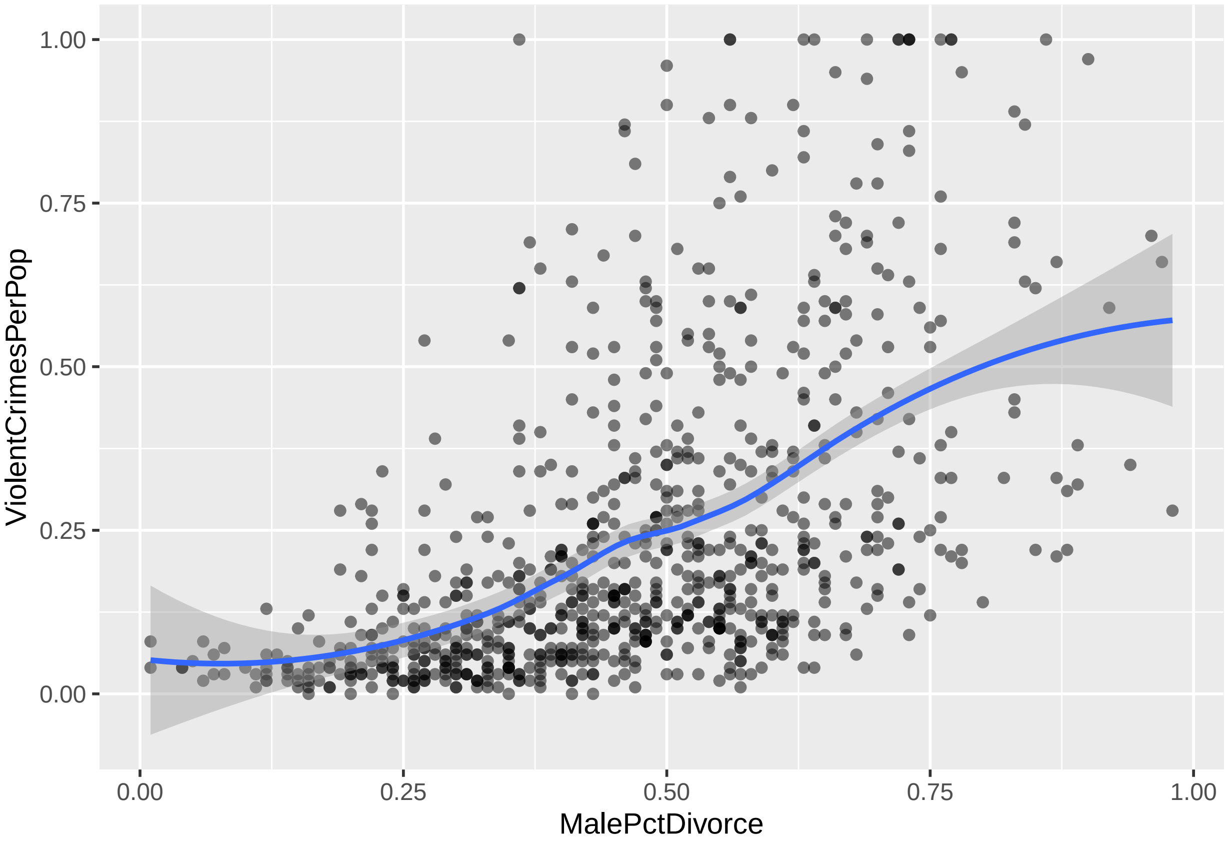


```
ggplot(data = d, mapping = aes(x = pctUrban, y = ViolentCrimesPerPop)) +
  geom_point(alpha = .5)
```



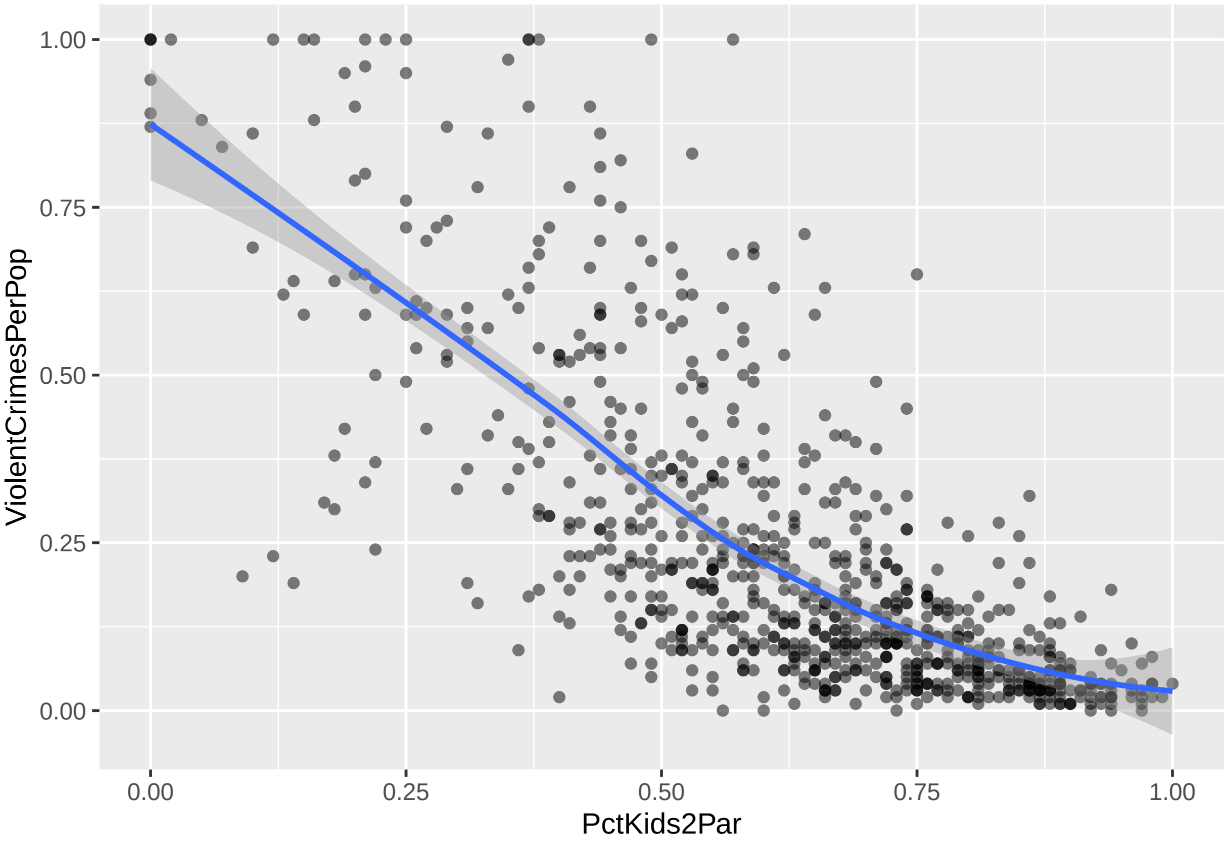
```r
ggplot(data = d, mapping = aes(x = MalePctDivorce, y = ViolentCrimesPerPop)) +
  geom_point(alpha = .5) +
  geom_smooth()
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



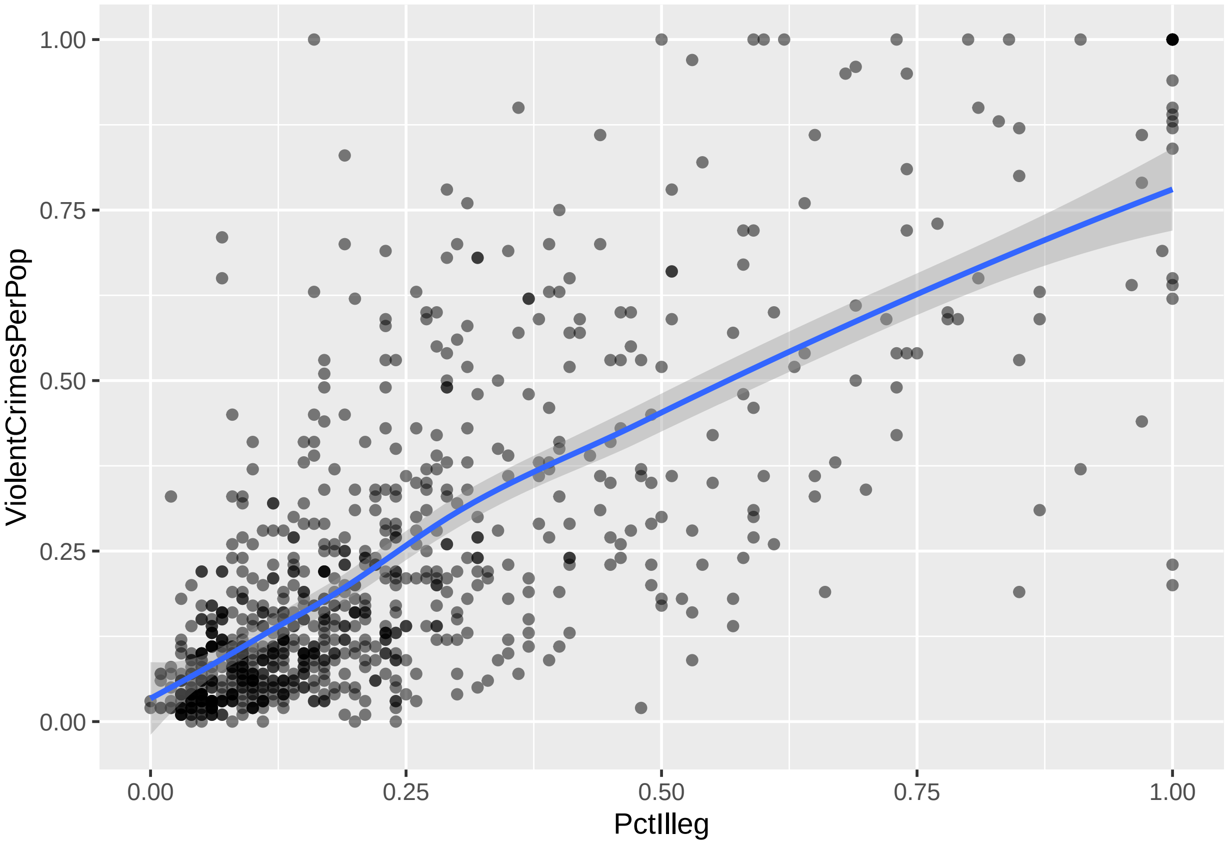
```
ggplot(data = d, mapping = aes(x = PctKids2Par, y = ViolentCrimesPerPop)) +
  geom_point(alpha = .5) +
  geom_smooth()
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



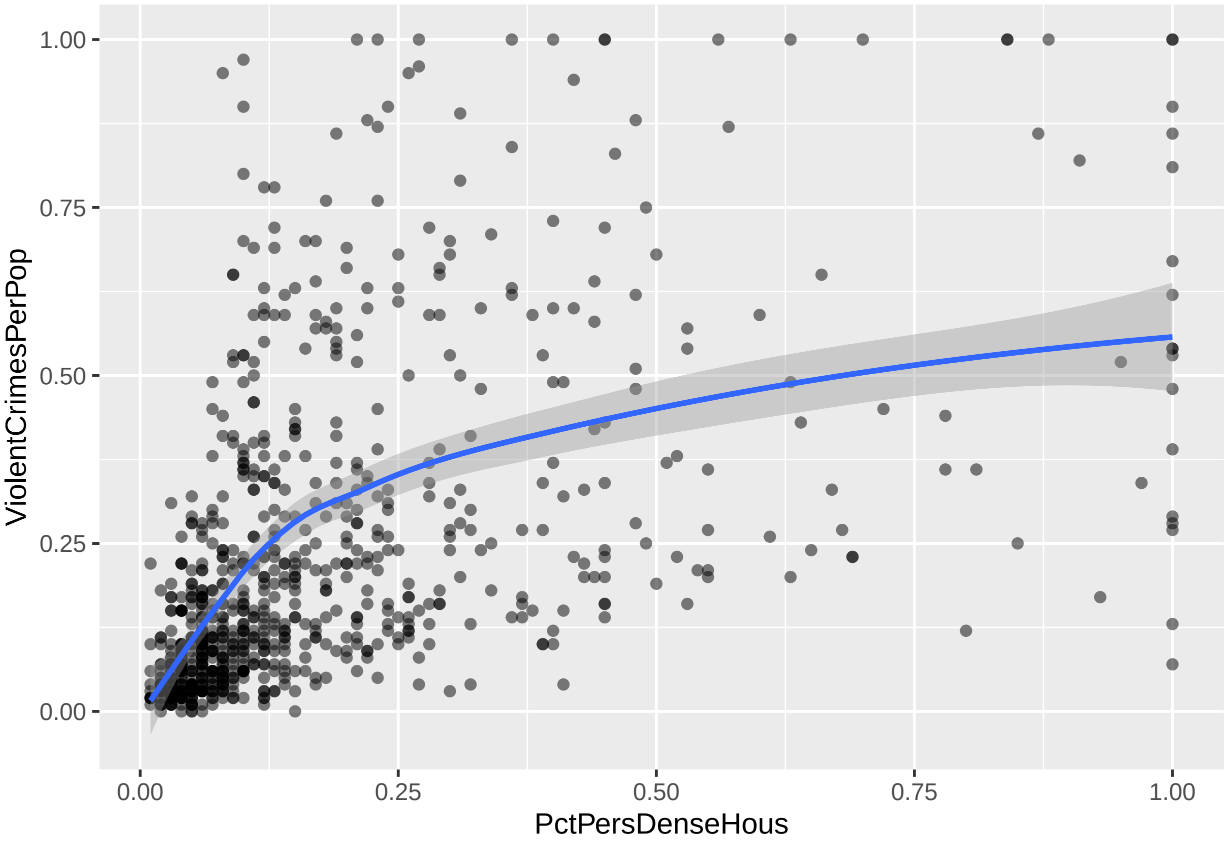
```
ggplot(data = d, mapping = aes(x = PctIlleg, y = ViolentCrimesPerPop)) +
  geom_point(alpha = .5) +
  geom_smooth()
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



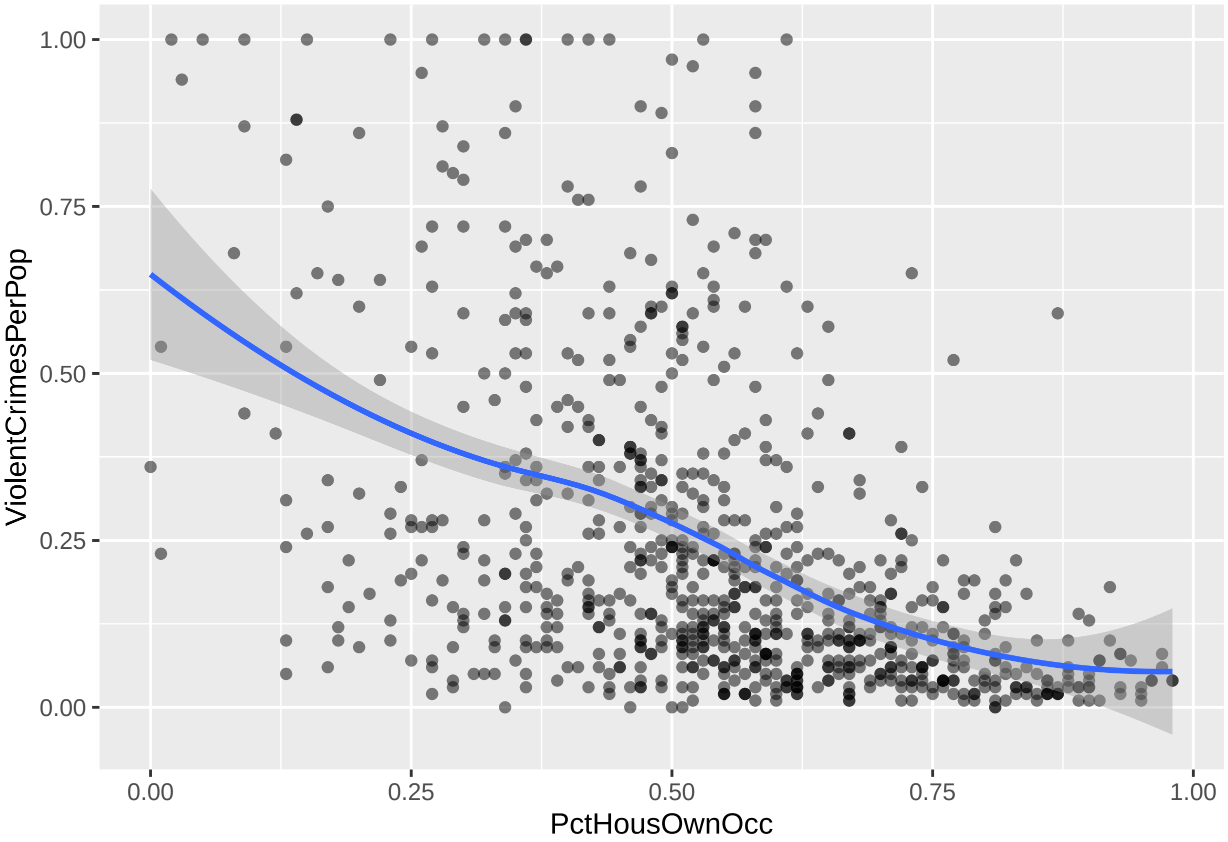
```r
ggplot(data = d, mapping = aes(x = PctPersDenseHous, y = ViolentCrimesPerPop)) +
  geom_point(alpha = .5) +
  geom_smooth()
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



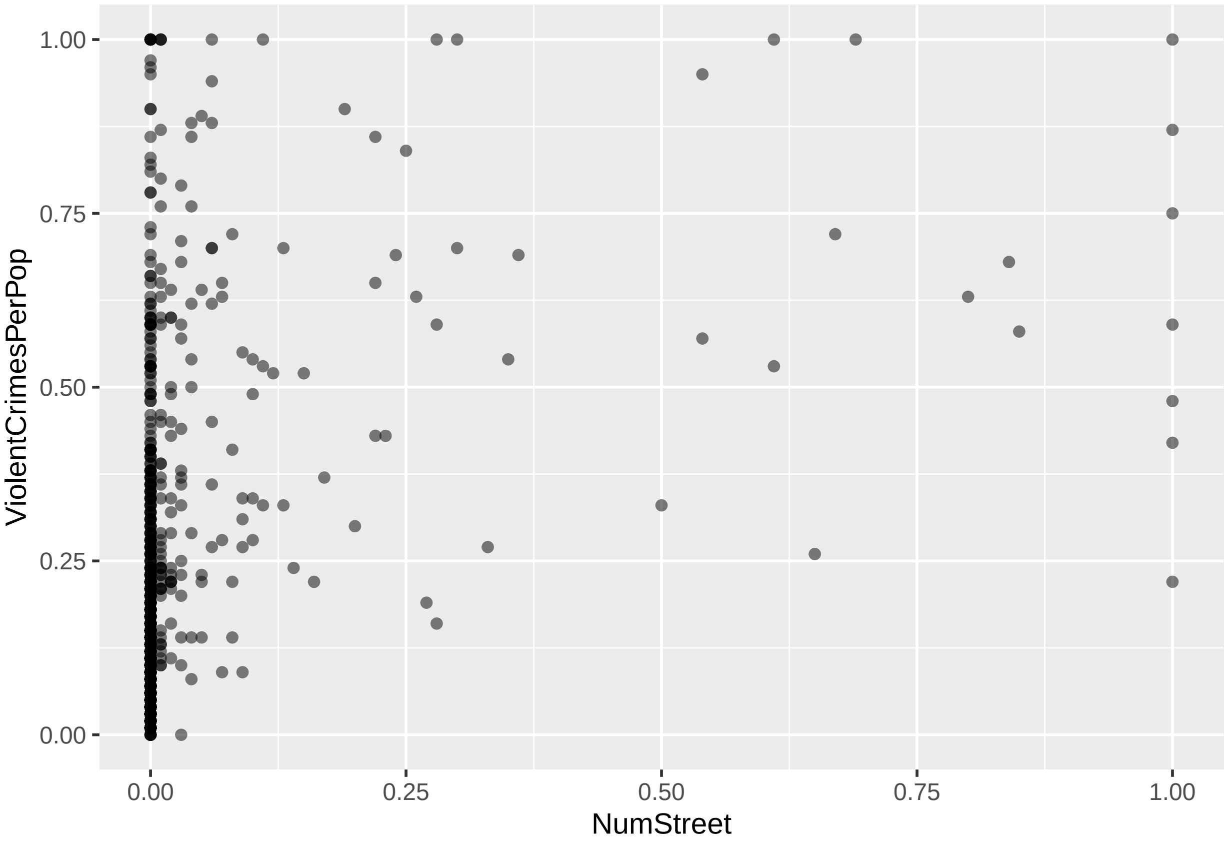
```
ggplot(data = d, mapping = aes(x = PctHousOwnOcc, y = ViolentCrimesPerPop)) +
  geom_point(alpha = .5) +
  geom_smooth()
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



```r
ggplot(data = d, mapping = aes(x = NumStreet, y = ViolentCrimesPerPop)) +
  geom_point(alpha = .5) +
  geom_smooth()
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =
## parametric, : at -0.005

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =
## parametric, : radius 2.5e-05

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =
## parametric, : all data on boundary of neighborhood. make span bigger

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =
## parametric, : pseudoinverse used at -0.005

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =
## parametric, : neighborhood radius 0.005

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =
## parametric, : reciprocal condition number 1

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =
## parametric, : zero-width neighborhood. make span bigger

## Warning: Computation failed in `stat_smooth()`:
## NA/NaN/Inf in foreign function call (arg 5)
```



```r
#I(MalePctDivorce^(1/3))+I(PctKids2Par^(1/3)) + PctIlleg+(PctPersDenseHous^2) + RentLowQ+MedRent+NumStr
```

```r
library(tidyverse)
library(leaps)

d <- read.csv("http://andrewpbray.github.io/data/crime-train.csv")

# Data wrangling
group_F_process <- function(training_data) {
  dw<-as_tibble(training_data) %>%
    mutate(
           MalePctDivorceCub = I(MalePctDivorce^(1/3)),
           PctKids2ParCub = I(PctKids2Par^(1/3)),
           PctPersDenseHousSq = PctPersDenseHous^2
           )
  dw<-select (d,-c(state,county,community,communityname,LemasSwornFT,LemasSwFTPerPop,LemasSwFTFieldOps,
  vars<-c()
  for (i in 1:(length(dw)-1)) {
    vars<- c(vars, names(dw)[i])
  }
  sqrd<-data.frame(lapply(vars, function(x){dw[,x]^(1/2)}))
  cubc<-data.frame(lapply(vars, function(x){dw[,x]^(1/3)}))
  names(sqrd)<-paste0(vars, "Sq")
  names(cubc)<-paste0(vars, "Cub")
  dw<-cbind(dw, sqrd,cubc)
  return(dw)
}
```

```r
# Manually fits model
group_F_fit <- function(training_data) {
 m1 <- lm(ViolentCrimesPerPop~
            MalePctDivorceCub+
            PctKids2ParCub+
            PctIlleg+
            PctPersDenseHousSq+
            RentLowQ+
            MedRent,
          training_data)
 m1
}

# Gets MSE
group_F_MSE <- function(model, data) {
  mean((data$ViolentCrimesPerPop - predict.lm(model, data)) ^ 2)
}

# Automatically fits model
group_F_automated_fit <- function(data, method) {
  leaps<-regsubsets(ViolentCrimesPerPop~.,
                    data = data,
                    nvmax = 25,
                    method = method)
  best<-summary(leaps)$which[which.max(summary(leaps)$adjr2),]
  variables <- c()
  for (i in 2:length(best)) {
    if (best[i] == TRUE) {
      variables <- c(variables, names(best)[i])
    }
  }
  vars<- paste(variables, collapse = "+")
  formula<- paste("lm(ViolentCrimesPerPop ~ ",vars,", data = dw)") # not able to get dataframe name as
  m1<-eval(parse(text=formula))
  return(m1)
}

dw <- group_F_process(d)
bestF <- group_F_automated_fit(dw, "forward")
```

```
## Warning in leaps.setup(x, y, wt = wt, nbest = nbest, nvmax = nvmax,
## force.in = force.in, : 1 linear dependencies found

## Reordering variables and trying again:
```

```r
bestB <- group_F_automated_fit(dw, "backward")
```

```
## Warning in leaps.setup(x, y, wt = wt, nbest = nbest, nvmax = nvmax,
## force.in = force.in, : 1 linear dependencies found

## Reordering variables and trying again:
```

```r
MSE_F <- group_F_MSE(bestF, dw)
MSE_B <- group_F_MSE(bestB, dw)
```