Lab 3: Regression Competition

In the hot New Jersey night

Alyssa Andrichik

---

partner: "Emmett Powers"" life saver: "Simon Couch" ##Lab 3: Regression Competition ###In the hot New Jersey night Your objective is to build a multiple regression model that predicts the amount of violent crime in a community as a function of other characteristics of those communities. You will construct your model using a training data set with information on over 100 variables recorded for 1000 communities. I've held back the data on 200 communities; this will server as the test data set for assessing the predictive accuracy of your model. Your model will be due by noon on Friday.

The main components of this lab are two R functions, described below, but consider and record the following as you go through the model building process.

```
d <- read.csv("http://andrewpbray.github.io/data/crime-train.csv")
library(MASS)
library(tidyverse)
library(broom)
library(dplyr)
```

1. Understanding the data provenance

A codebook for the data can be found here. Read through that and jot down any predictors that you expect to have a strong association with violent crime - these will serve as a good jumping off point. You'll notice there are some variables that have lots of missing data. I recommend that you don't include these in your model.

```
d[d=="?"]<-NA
d[d==""]<-NA
```

```
goodershit <- d %>%
  select(-c(state, county, community, communityname, population, LemasSwornFT, LemasSwFTPerPop,
            LemasSwFTFieldOps, LemasSwFTFieldPerPop, LemasTotalReq, LemasTotReqPerPop, PolicReqPerOffic
            PolicPerPop, RacialMatchCommPol, PctPolicWhite, PctPolicBlack, PctPolicHisp, PctPolicAsian,
            PctPolicMinor, OfficAssgnDrugUnits, NumKindsDrugsSeiz, PolicAveOTWorked, PolicCars, PolicOp
            LemasPctPolicOnPatr, LemasGangUnitDeploy, PolicBudgPerPop)) %>%
  drop_na()
view(goodershit)

realgoodshit <- goodershit %>%
  cor(goodershit)
```

Correlation of variables with ViolentCrimesPerPop

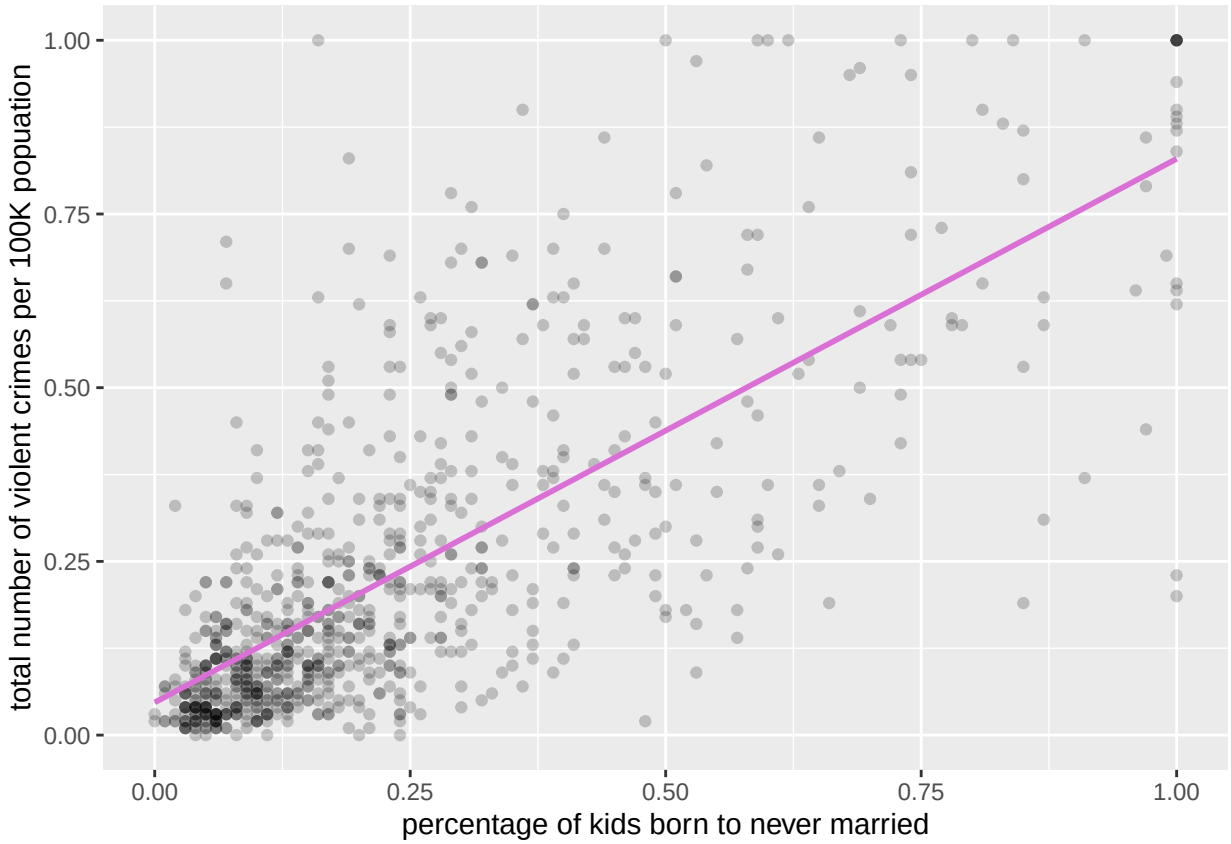
PctIlleg: percentage of kids born to never married – 0.74352441 racepctblack: percentage of population that is african american – 0. 61396953 PctKids2Par: percentage of kids in family housing with two parents – -0.73929315 PctFam2Par: percentage of families (with kids) that are headed by two parents – -0.70528060 racePctWhite: percentage of population that is caucasian – -0.68885560 PctYoungKids2Par: percent of kids 4 and under in two parent households – -0.66897706 PctTeen2Par: percent of kids age 12-17 in two parent

households – -0.65329249 PctPersOwnOccup: percent of people in owner occupied households – -0.57620572 pctWPubAsst: percentage of households with public assistance income in 1989 – 0.59227263

2. Exploratory data analysis

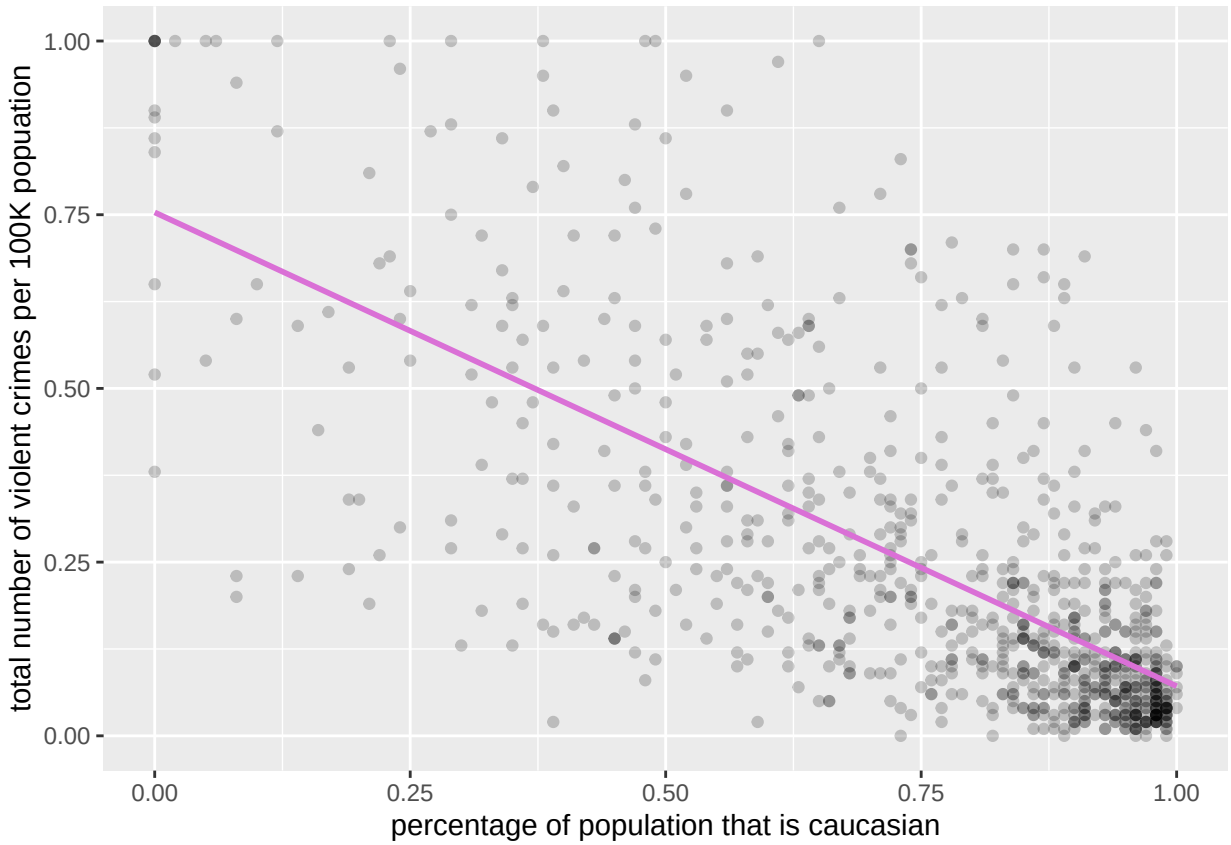
This data set is far too large to make a full pairs plot, so start off looking at the pairwise relationships between the response and the predictors you expect will be useful. Do these relationships look strong/weak? Linear/non-linear? Does it seem like a transformation would be useful? To read in the data, you can use

```
ggplot(data = goodershit, mapping = aes(x = PctIlleg,
y = ViolentCrimesPerPop)) +
geom_point(alpha = 0.2) +
  labs(x = "percentage of kids born to never married",
       y = "total number of violent crimes per 100K popuation")+
  geom_smooth(method = "lm", se = FALSE, col = "orchid")
```



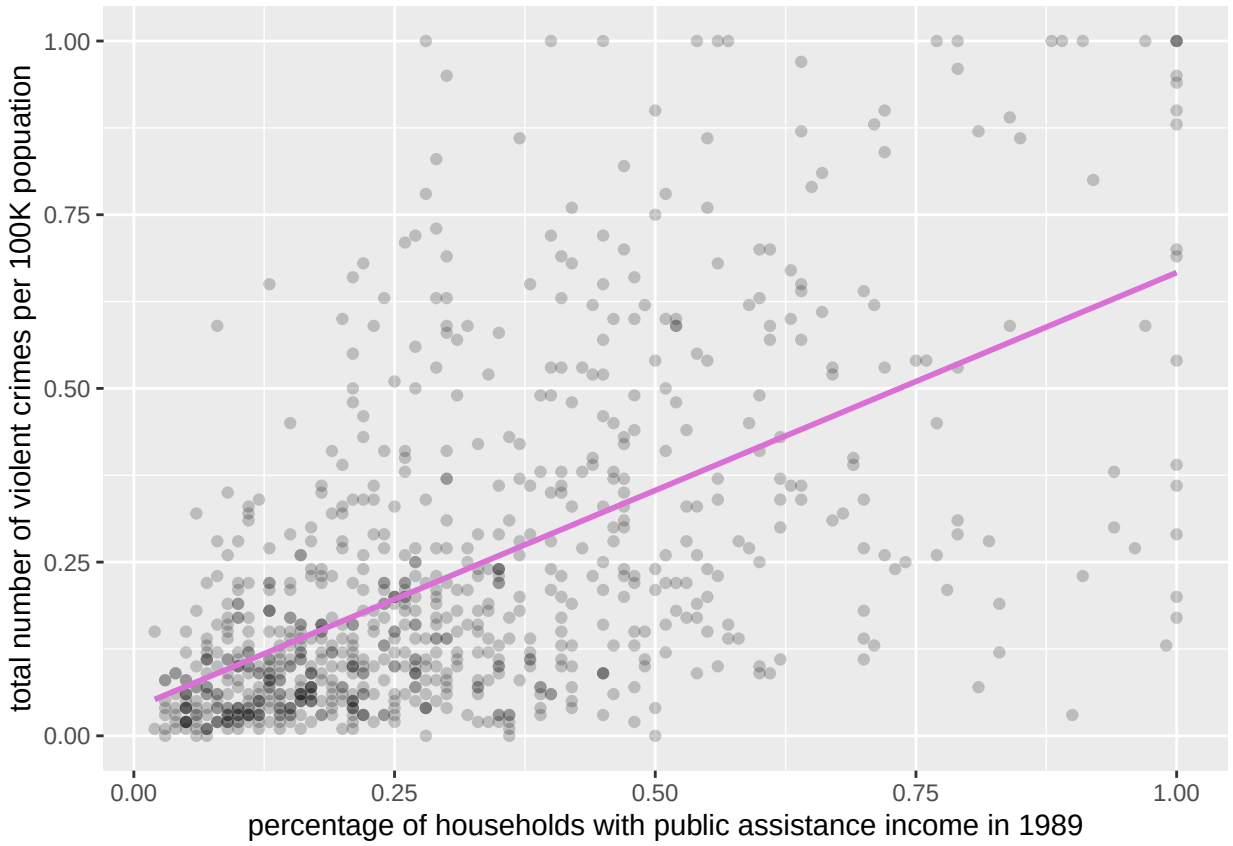
Correlation Coefficent: 0.74352441

```
ggplot(data = goodershit, mapping = aes(x = racePctWhite,
y = ViolentCrimesPerPop)) +
geom_point(alpha = 0.2) +
  labs(x = "percentage of population that is caucasian",
       y = "total number of violent crimes per 100K popuation")+
  geom_smooth(method = "lm", se = FALSE, col = "orchid")
```



Correlation Coefficent: -0.68885560

```
ggplot(data = goodershit, mapping = aes(x = pctWPubAsst,
y = ViolentCrimesPerPop)) +
geom_point(alpha = 0.2) +
  labs(x = "percentage of households with public assistance income in 1989",
       y = "total number of violent crimes per 100K popuation")+
  geom_smooth(method = "lm", se = FALSE, col = "orchid")
```

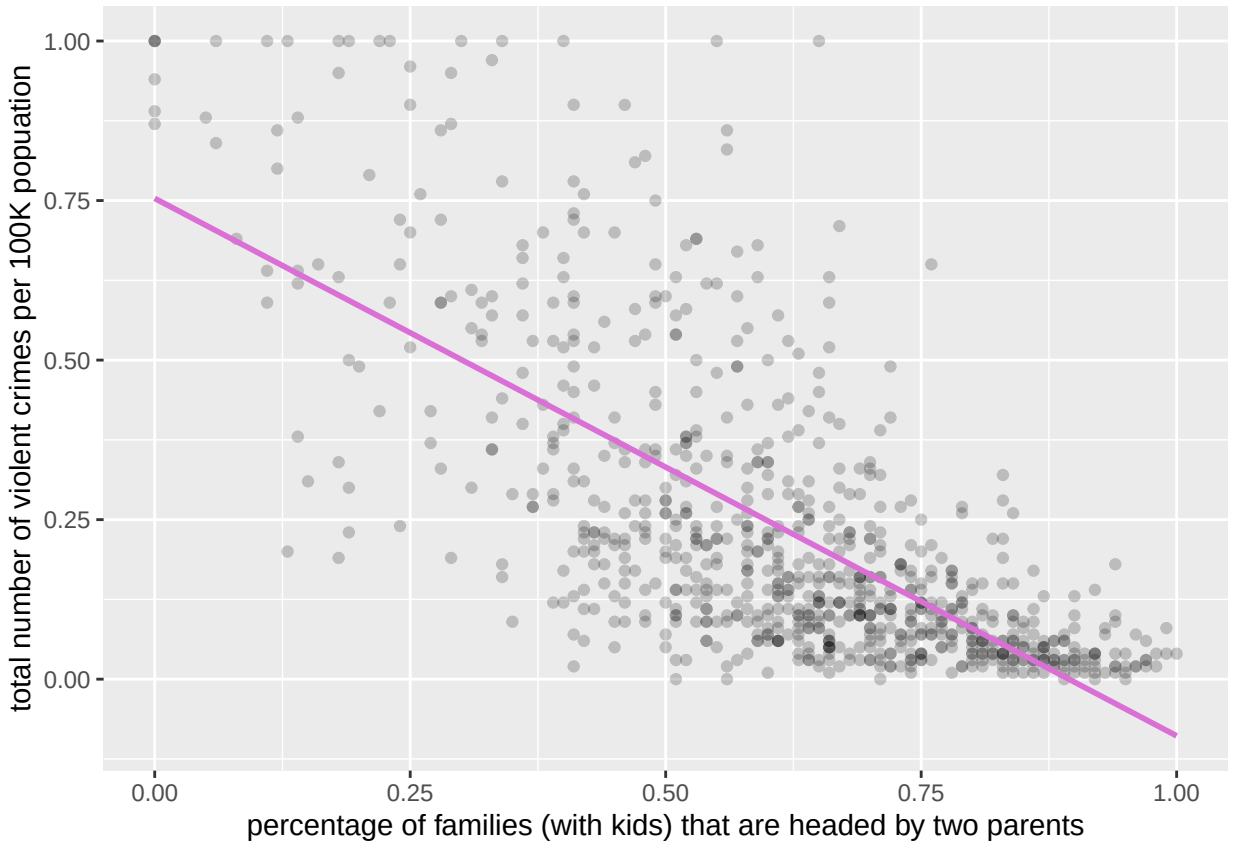


Correlation Coefficent: 0.59227263

```
ggplot(data = goodershit, mapping = aes(x = PctPersOwnOccup,
y = ViolentCrimesPerPop)) +
geom_point(alpha = 0.2) +
  labs(x = "percent of people in owner occupied households",
       y = "total number of violent crimes per 100K popuation")+
  geom_smooth(method = "lm", se = FALSE, col = "orchid")
```

Correlation Coefficent: -0.57620572

```
ggplot(data = goodershit, mapping = aes(x = PctFam2Par,
y = ViolentCrimesPerPop)) +
geom_point(alpha = 0.2) +
  labs(x = "percentage of families (with kids) that are headed by two parents",
       y = "total number of violent crimes per 100K popuation")+
  geom_smooth(method = "lm", se = FALSE, col = "orchid")
```



Correlation Coefficent: -0.70528060

3. Model building

Start building a series of models of varying complexity. Things to consider include transformations, polynomials, and interaction terms. You can compare predictive power of candidate models using several tools including statistics that penalize for complexity ((R^2_{adj}) and RSE) as well as the F-test (discussed in Ch. 3 of the book). Also, if there is multicollinearity between your predictors, that's a sign that some of them can be removed.

```
m0<-lm(ViolentCrimesPerPop ~ PctFam2Par + racePctWhite + PctPersOwnOccup, data = goodershit)
summary(m0)$r.squared
```

```
## [1] 0.5966378
```

```
mPctIlleg <- lm(ViolentCrimesPerPop ~ PctIlleg, data = goodershit)
summary(mPctIlleg)$r.squared
```

```
## [1] 0.5528286
```

```
mracePctWhite <- lm(ViolentCrimesPerPop ~ racePctWhite, data = goodershit)
summary(mracePctWhite)$r.squared
```

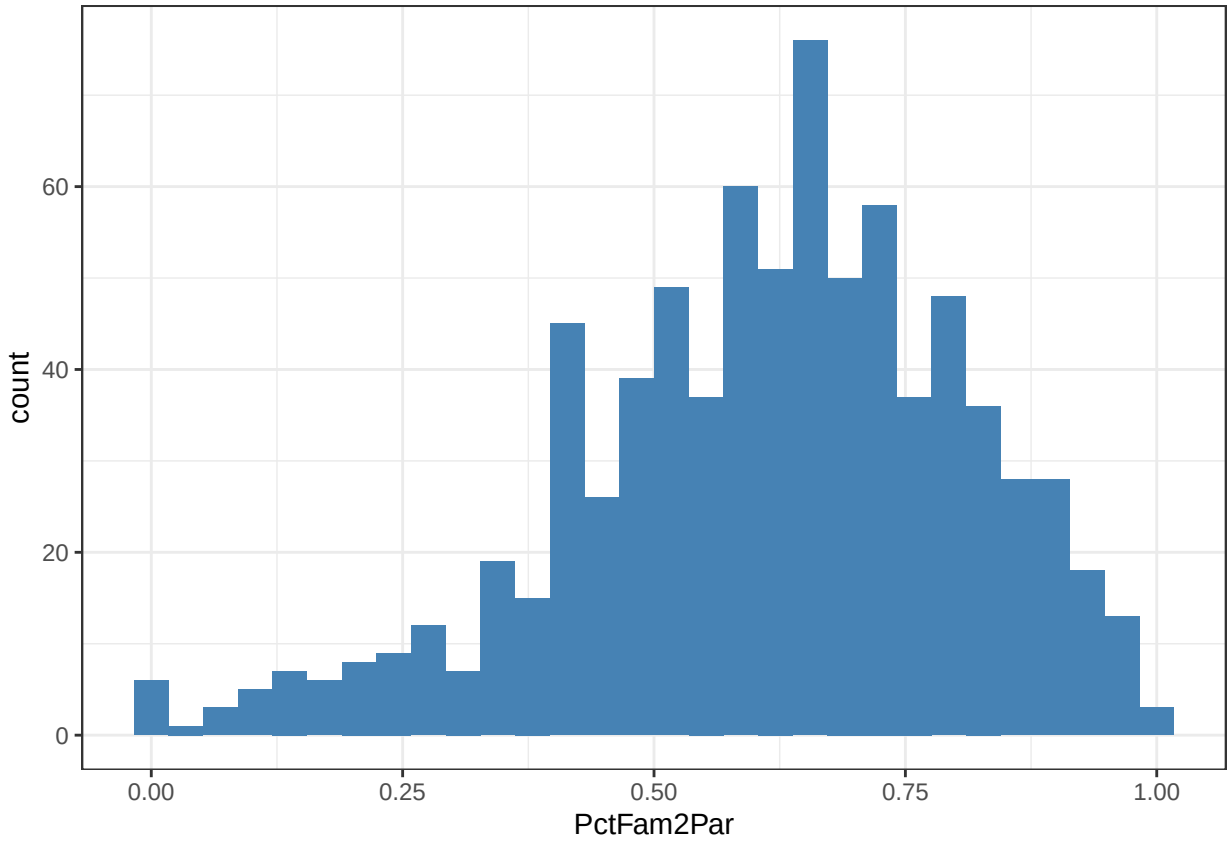
```
## [1] 0.474522
```

```
mPctPersOwnOccup <- lm(ViolentCrimesPerPop ~ pctWPubAsst, data = goodershit)
summary(mPctPersOwnOccup)$r.squared
```

```
## [1] 0.3507869
```

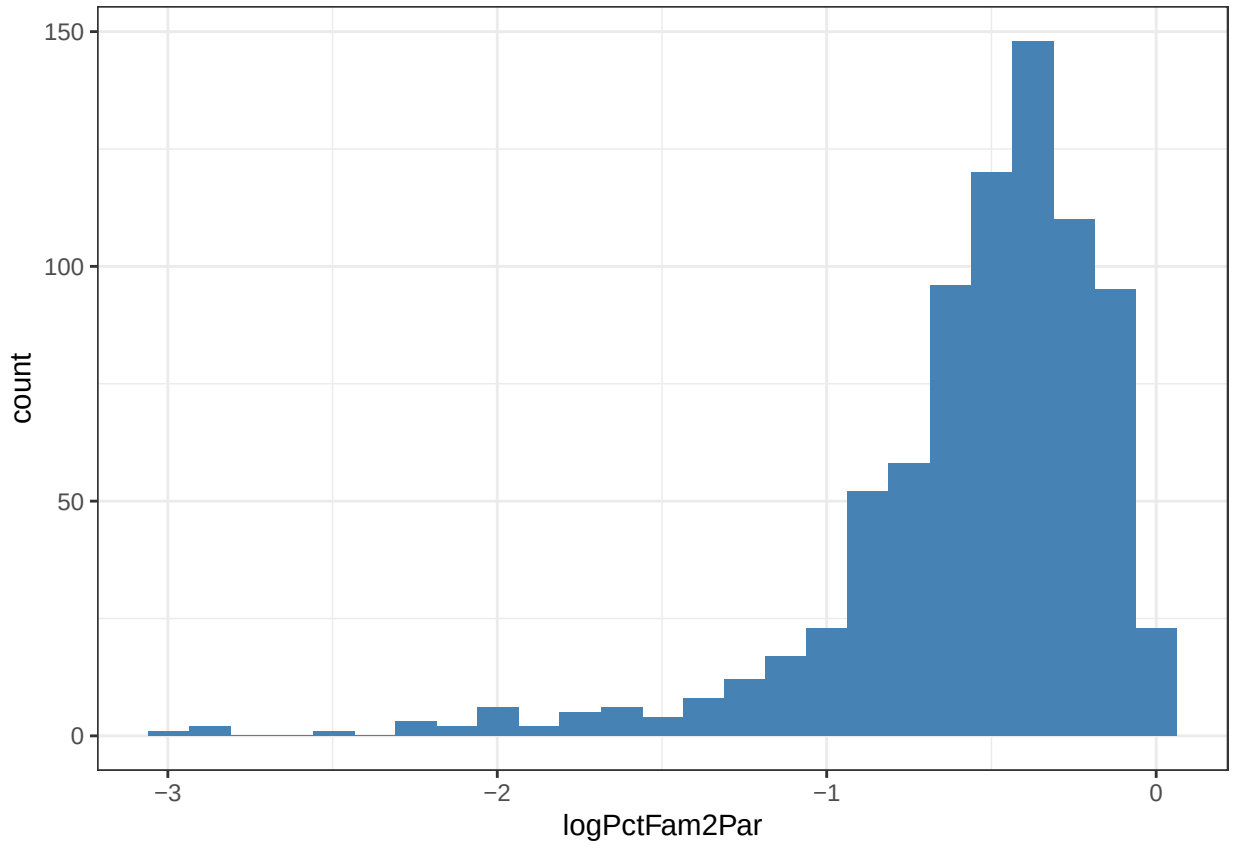
```
ggplot(goodershit, aes(x = PctFam2Par)) +
  geom_histogram(fill = "steelblue") +
  theme_bw()
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
logPctFam2Par <- log(goodershit$PctFam2Par)
ggplot(goodershit, aes(x = logPctFam2Par)) +
  geom_histogram(fill = "steelblue", bins = 25) +
  theme_bw()
```

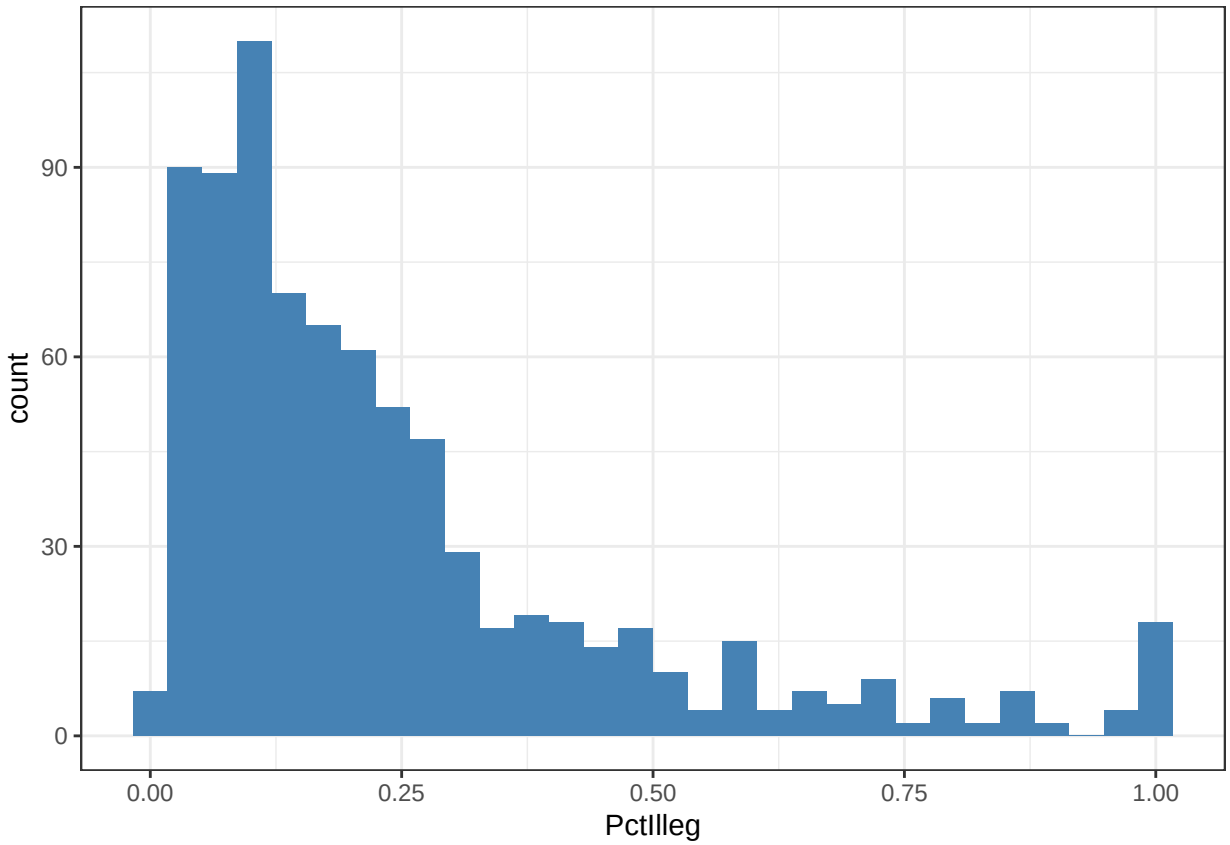
```
## Warning: Removed 6 rows containing non-finite values (stat_bin).
```



The log is less normal.

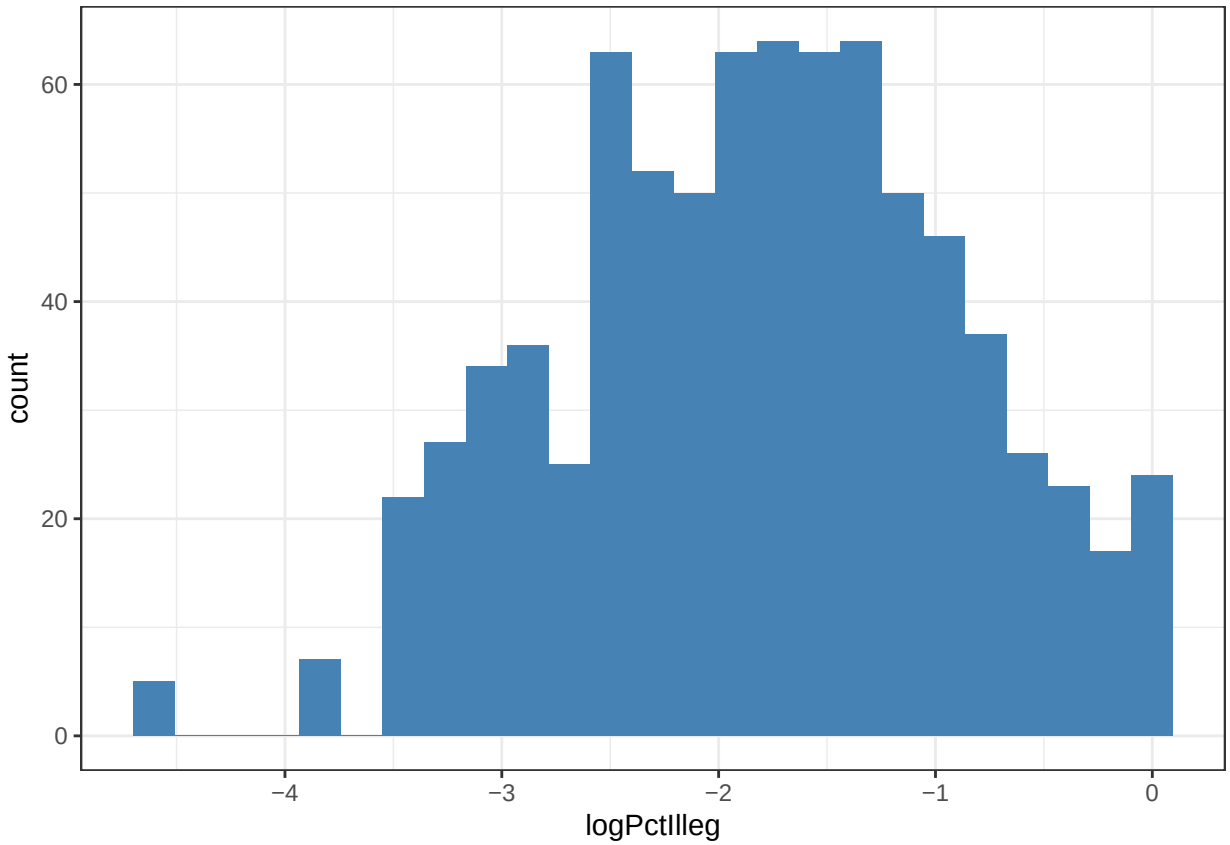
```
ggplot(goodershit, aes(x = PctIlleg)) +
  geom_histogram(fill = "steelblue") +
  theme_bw()
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
logPctIlleg <- log(goodershit$PctIlleg)
ggplot(goodershit, aes(x = logPctIlleg)) +
  geom_histogram(fill = "steelblue", bins = 25) +
  theme_bw()
```

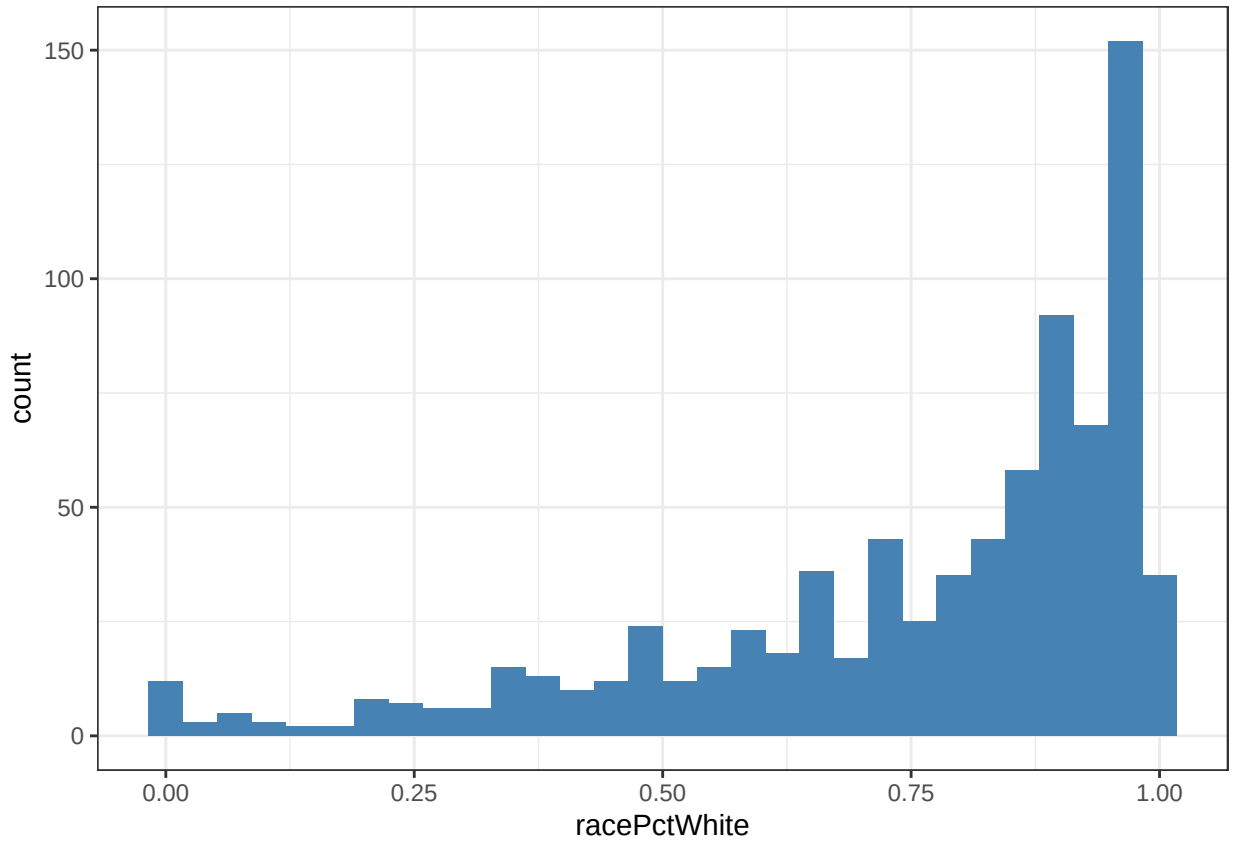
```
## Warning: Removed 2 rows containing non-finite values (stat_bin).
```



The log of PctIlleg has a far more normal distribution.

```
ggplot(goodershit, aes(x = racePctWhite)) +
  geom_histogram(fill = "steelblue") +
  theme_bw()
```

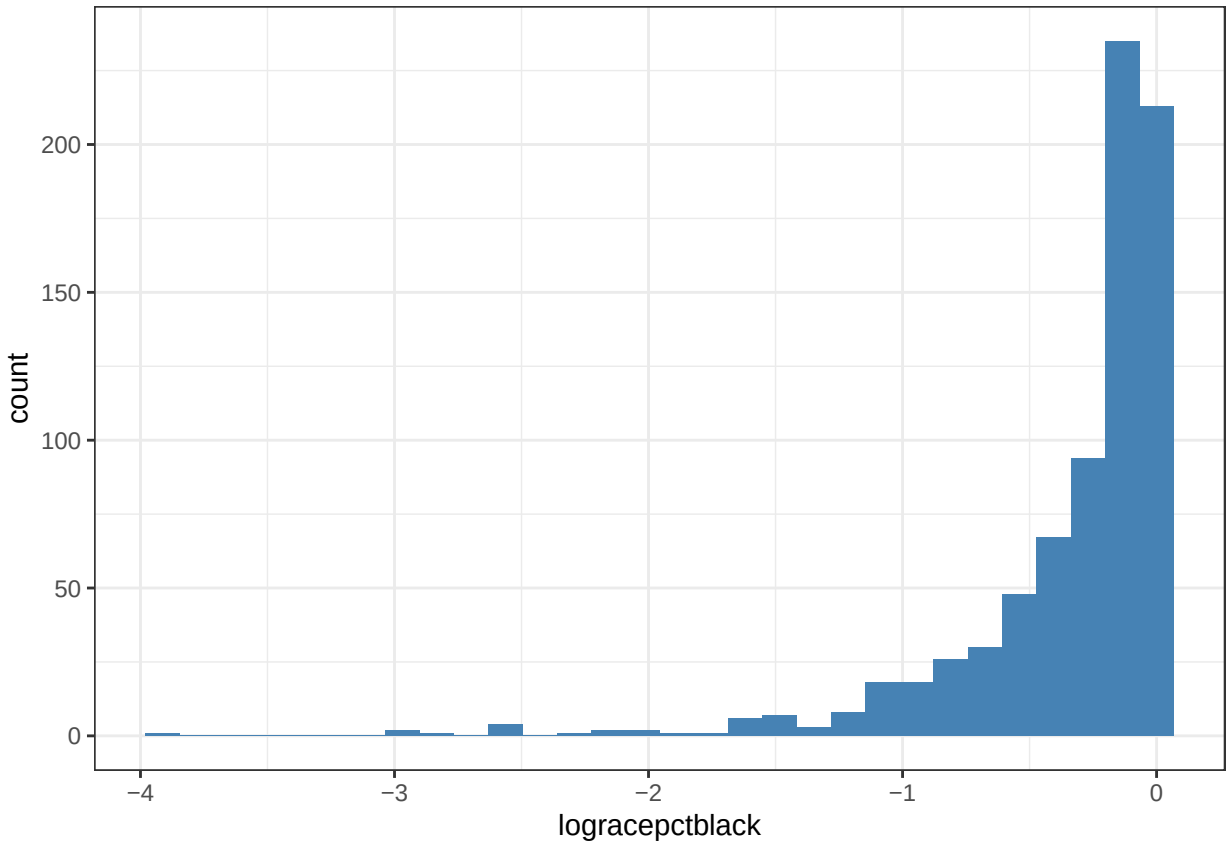
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
logracepctblack <- log(goodershit$racePctWhite)
ggplot(goodershit, aes(x = logracepctblack)) +
  geom_histogram(fill = "steelblue") +
  theme_bw()
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

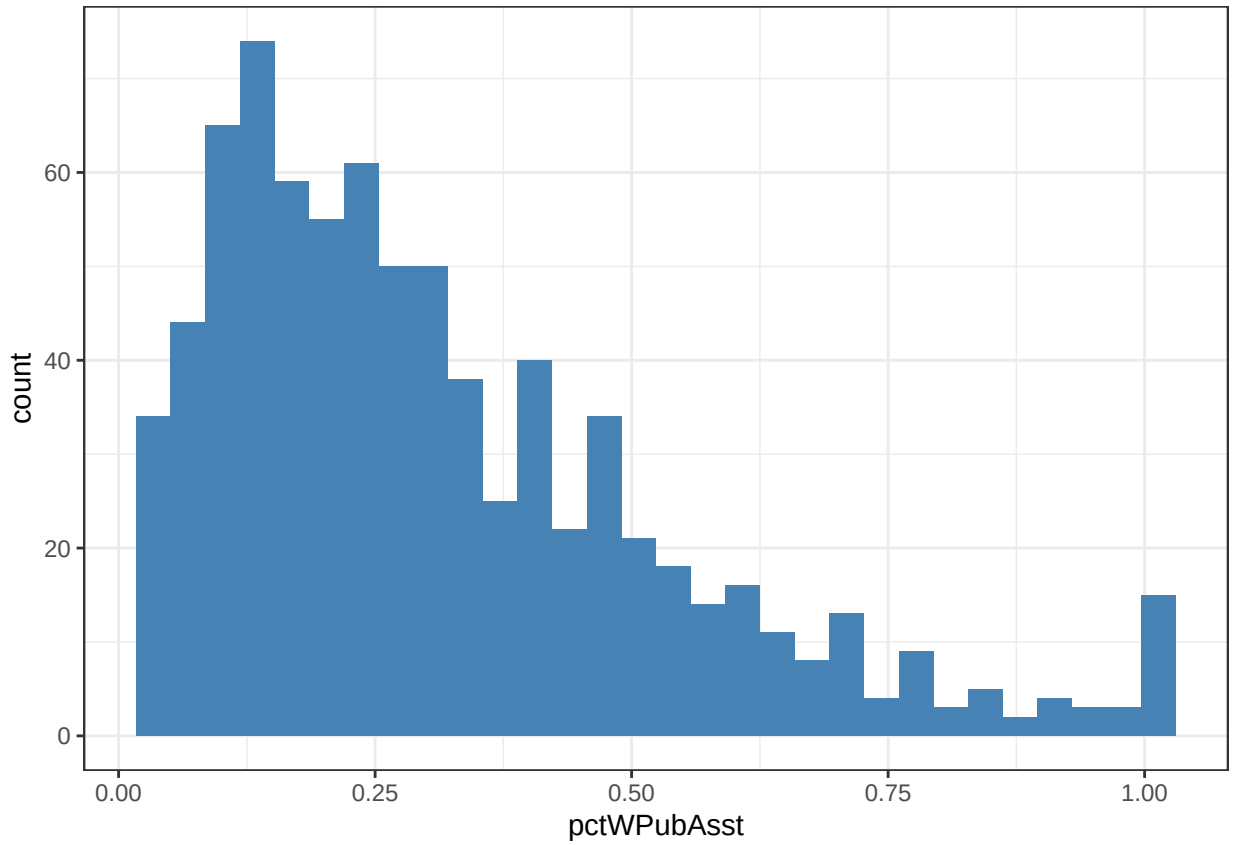
```
## Warning: Removed 12 rows containing non-finite values (stat_bin).
```



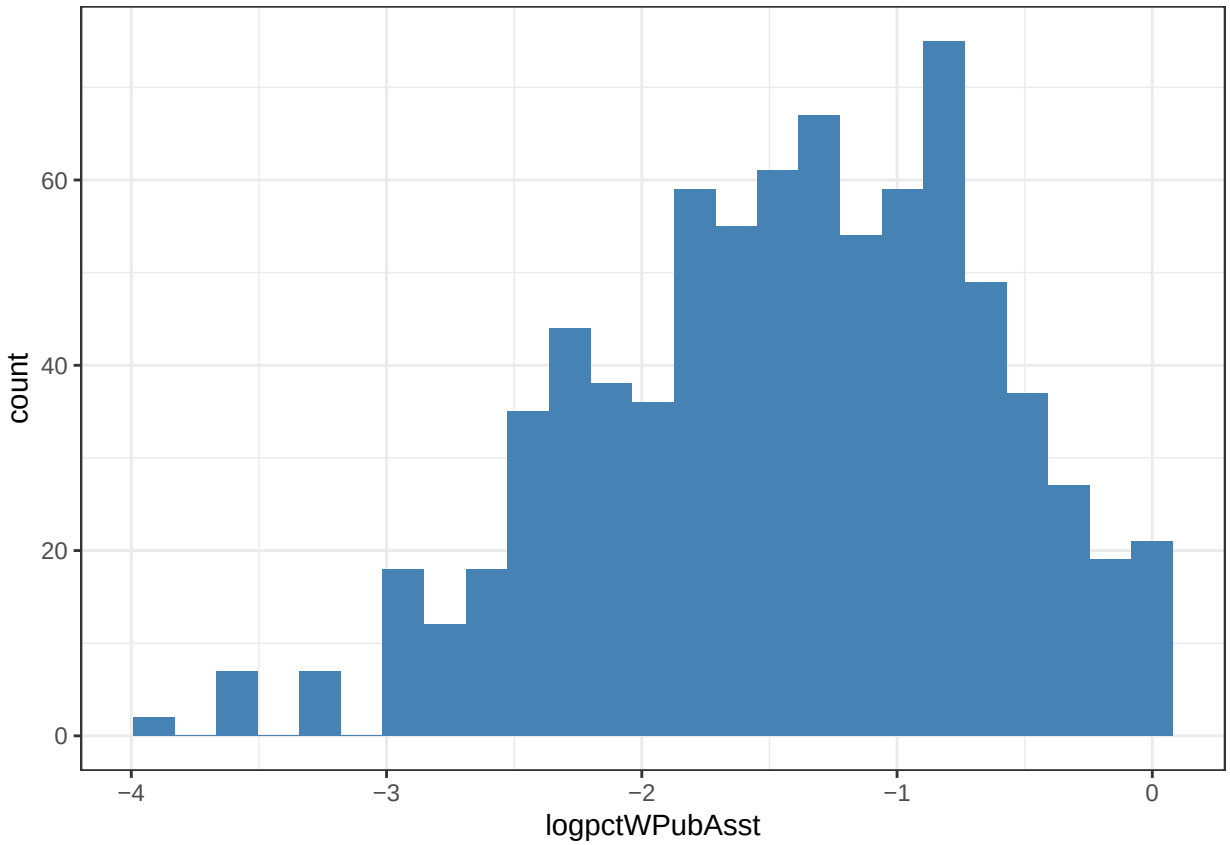
Not a normal distribution at all for either.

```
ggplot(goodershit, aes(x = pctWPubAsst)) +
  geom_histogram(fill = "steelblue") +
  theme_bw()
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

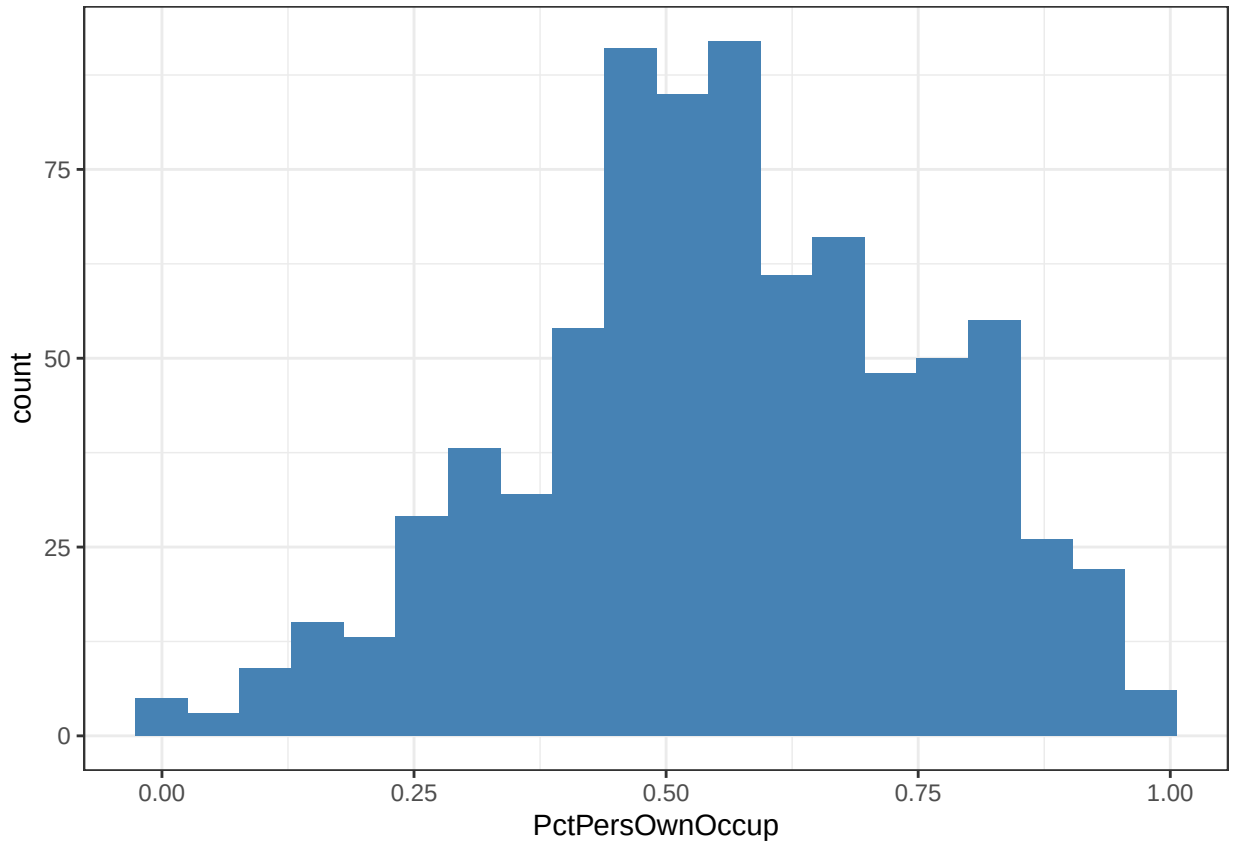



```
logpctWPubAsst <- log(goodershit$pctWPubAsst)
ggplot(goodershit, aes(x = logpctWPubAsst)) +
  geom_histogram(fill = "steelblue", bins = 25) +
  theme_bw()
```



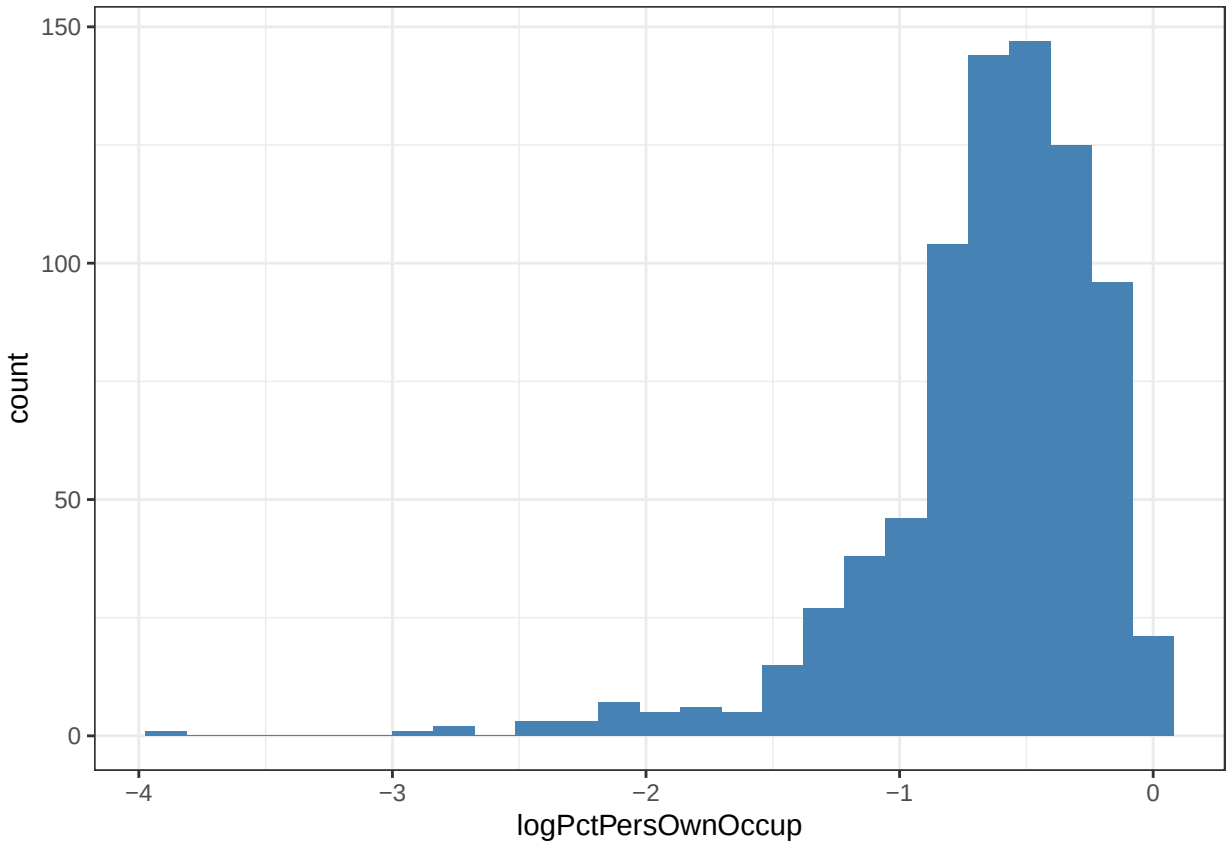
The log has a far more normal distribution.

```
ggplot(goodershit, aes(x = PctPersOwnOccup)) +
  geom_histogram(fill = "steelblue", bins = 20) +
  theme_bw()
```




```
logPctPersOwnOccup <- log(goodershit$PctPersOwnOccup)
ggplot(goodershit, aes(x = logPctPersOwnOccup)) +
  geom_histogram(fill = "steelblue", bins = 25) +
  theme_bw()
```

```
## Warning: Removed 4 rows containing non-finite values (stat_bin).
```

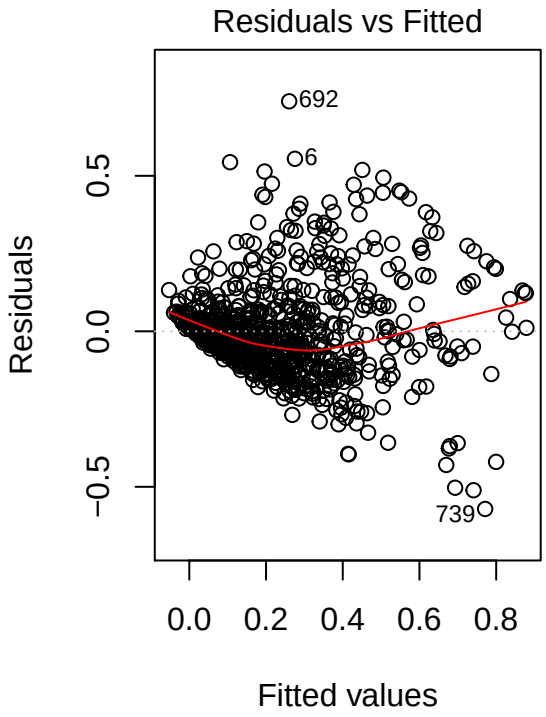


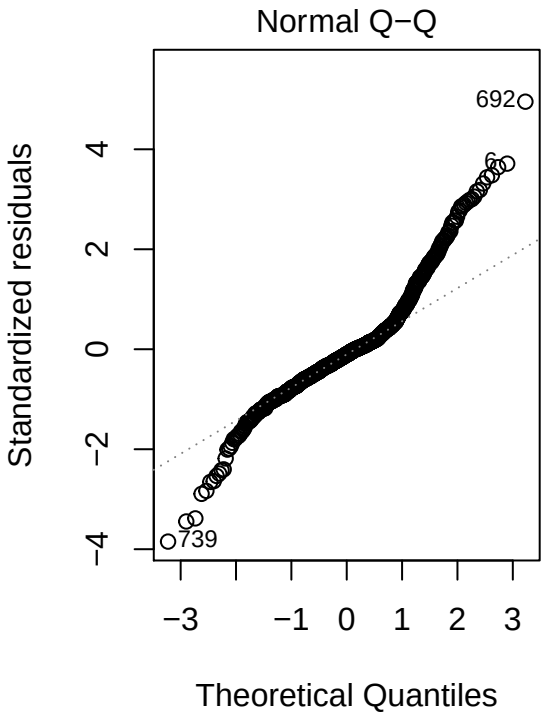
The original variable has a much more normal distribution than the log.

4. Model diagnostics

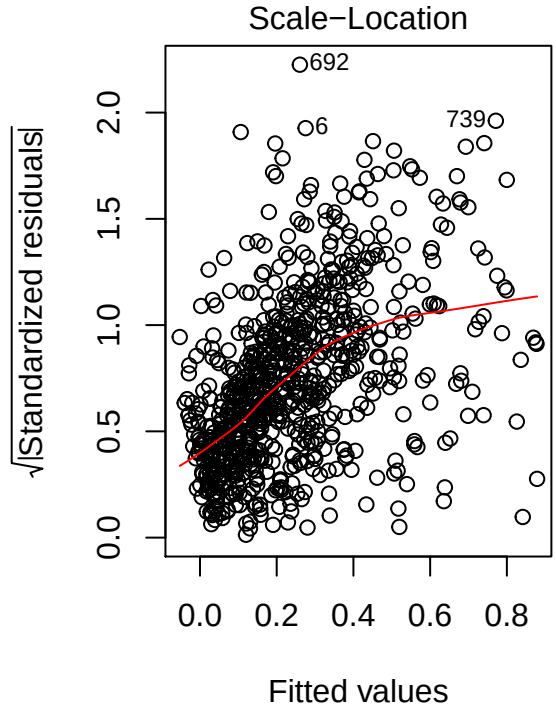
If you're relying on p-values to decide which variables to include in your model, you'll want to verify that they're accurate. Bring up residual plots for your models to assess the validity of your model (normal residuals with constant variance, linear trend).

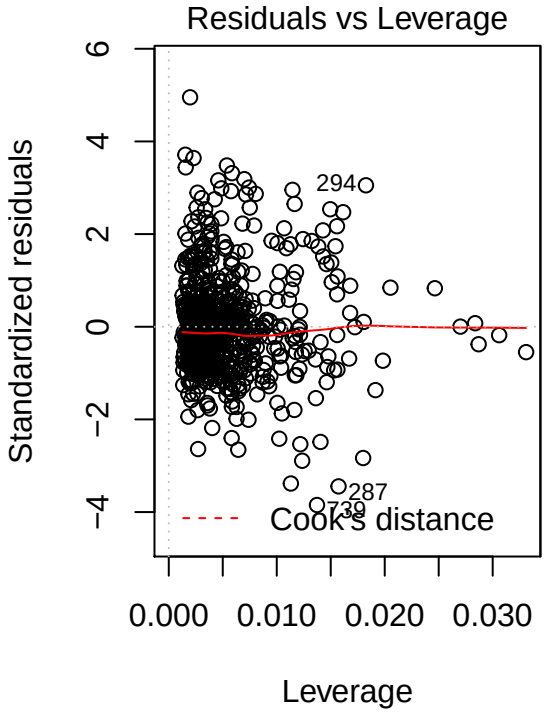
```
par(mfrow = c(1, 2))
plot(m0, 1:2)
```



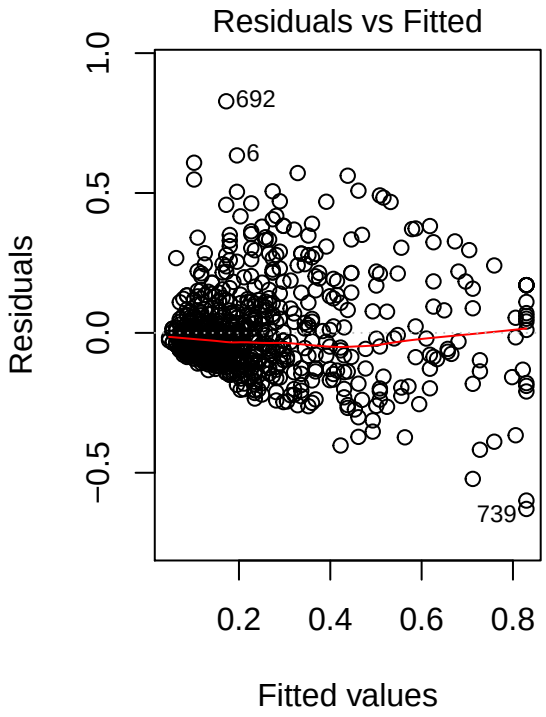


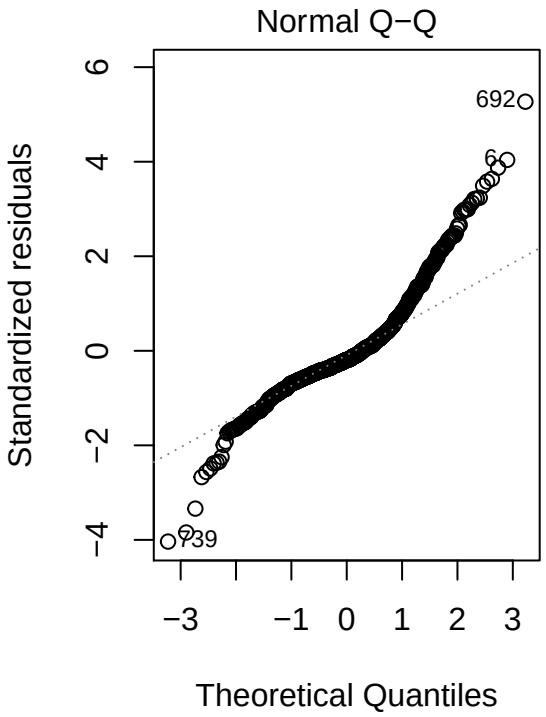

```
plot(m0, c(3, 5))
```



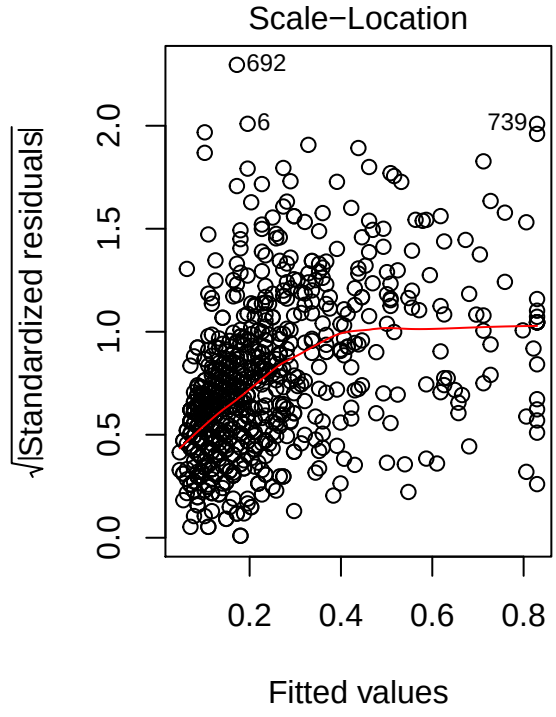


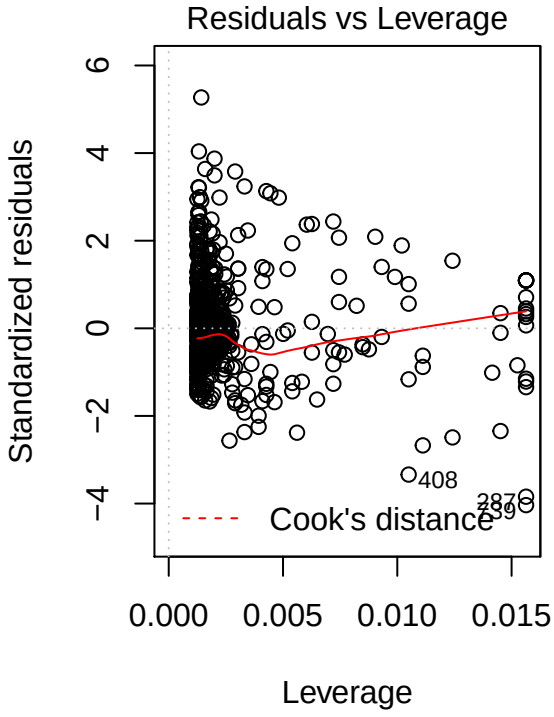

```
plot(mPctIlleg, 1:2)
```



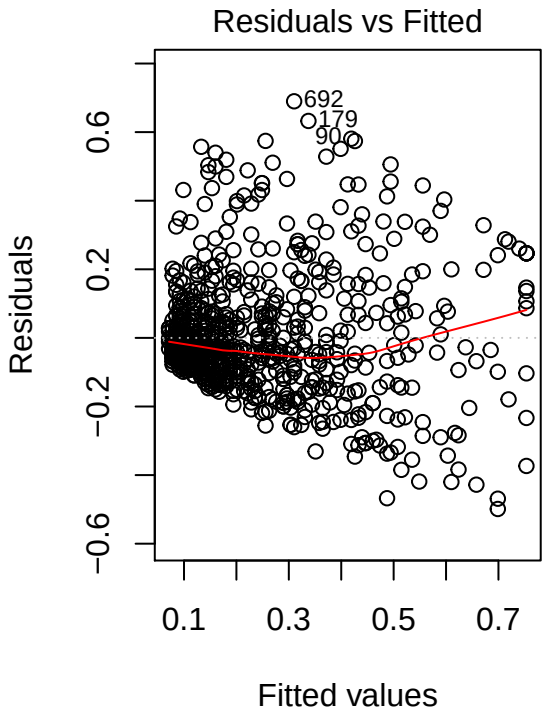


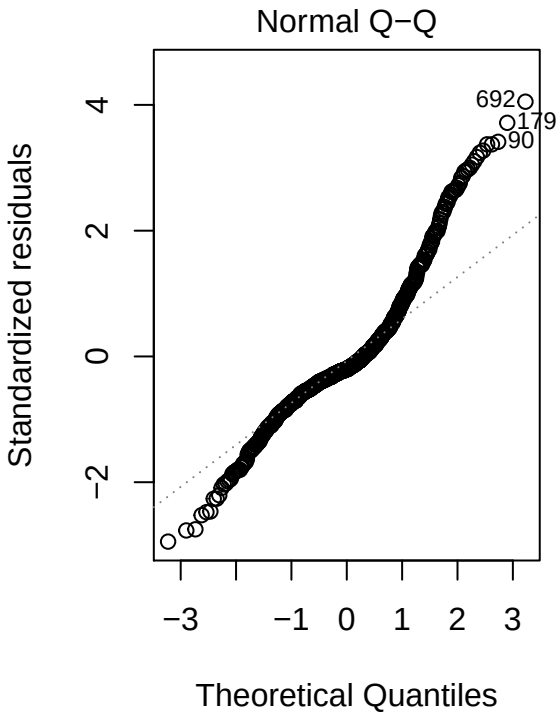

```
plot(mPctIlleg, c(3, 5))
```



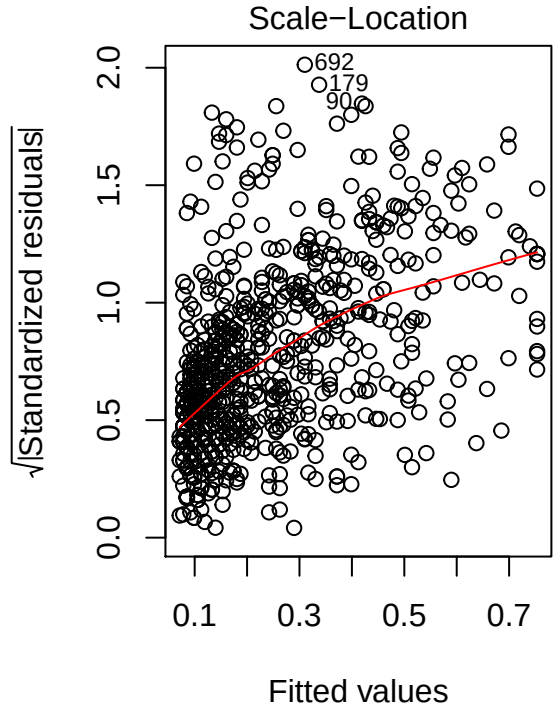


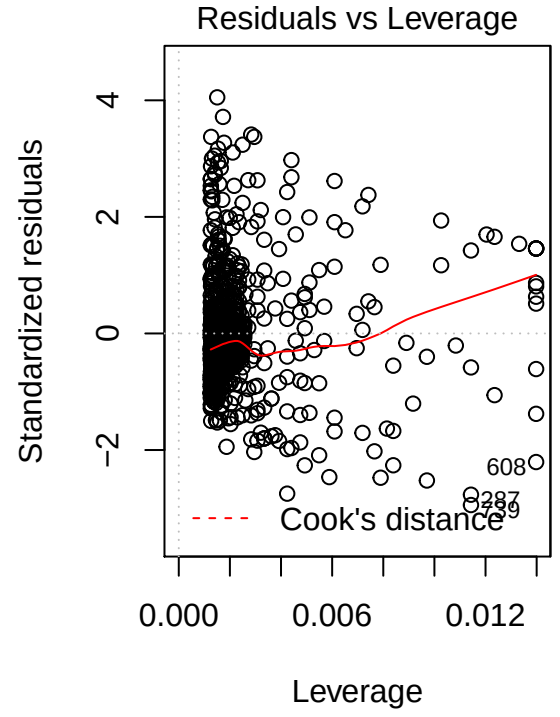

```
plot(mracePctWhite, 1:2)
```



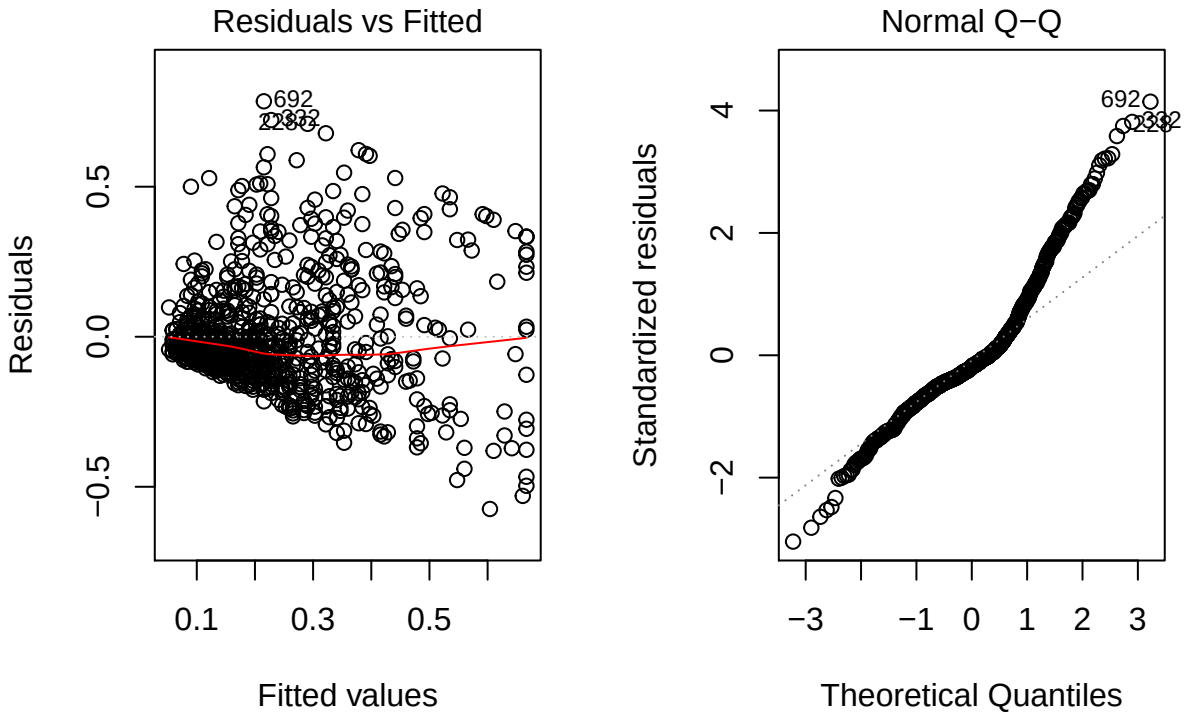



```
plot(mracePctWhite, c(3, 5))
```

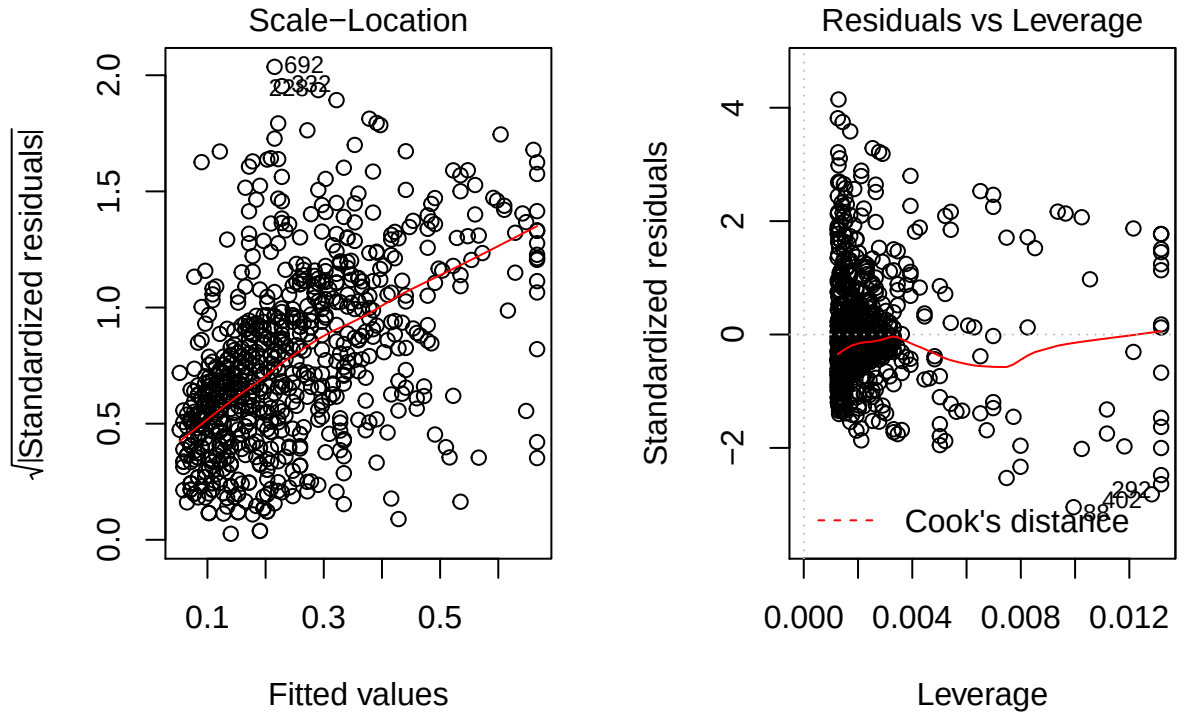





```
plot(mPctPersOwnOccup, 1:2)
```




```
plot(mPctPersOwnOccup, c(3, 5))
```



R Functions 1. Fitting the model Please write a function that takes the training data as input and then outputs an lm object. It should be of the form,

```
group_K_fit <- function(training_data) {
  library(tidyverse)
```

```
x <- sample(c(TRUE, FALSE), 800, replace = TRUE)
  train <-d[!x,]
  test <-d[x,]
  m1<-lm(ViolentCrimesPerPop ~ PctFam2Par + racePctWhite + PctPersOwnOccup, data = train)
}
```

2. Computing MSE Please write a function that takes as input the output from group_A_fit and a data set, and returns the MSE. (hint: there is a predict() function that may be helpful.)

```
group_K_MSE <- function(model, data) {
predict(m1, newdata = test)
training_MSE <- mean(m1$residuals^2)
training_MSE
}
```