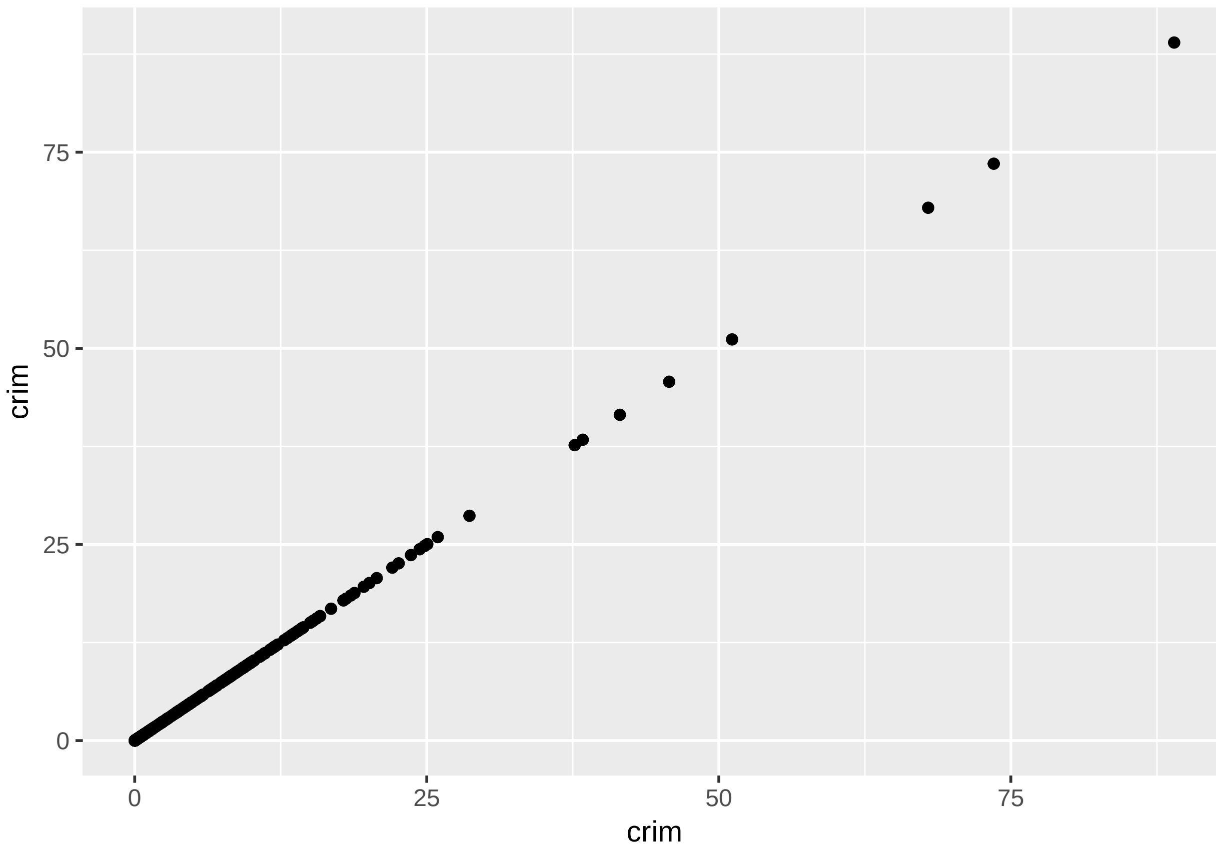
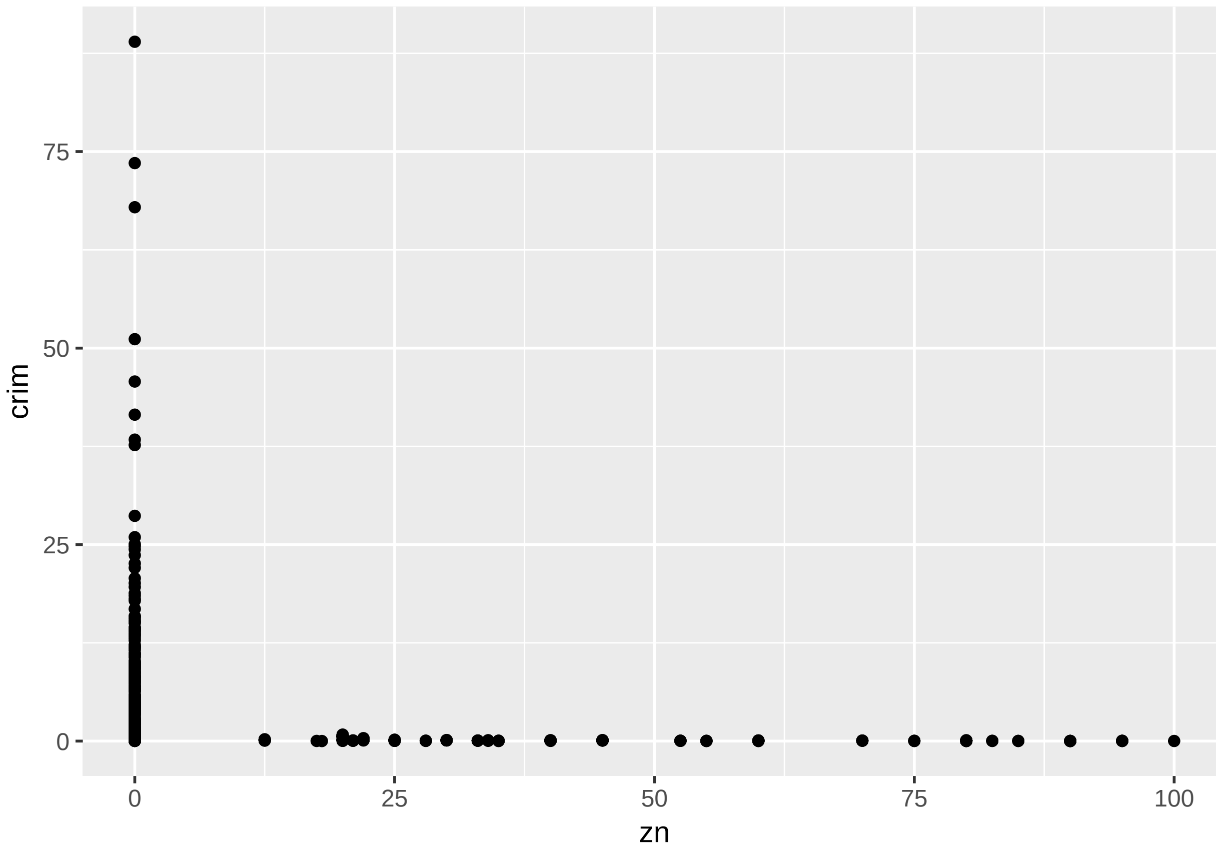
HW3-Zhenyu-Hong

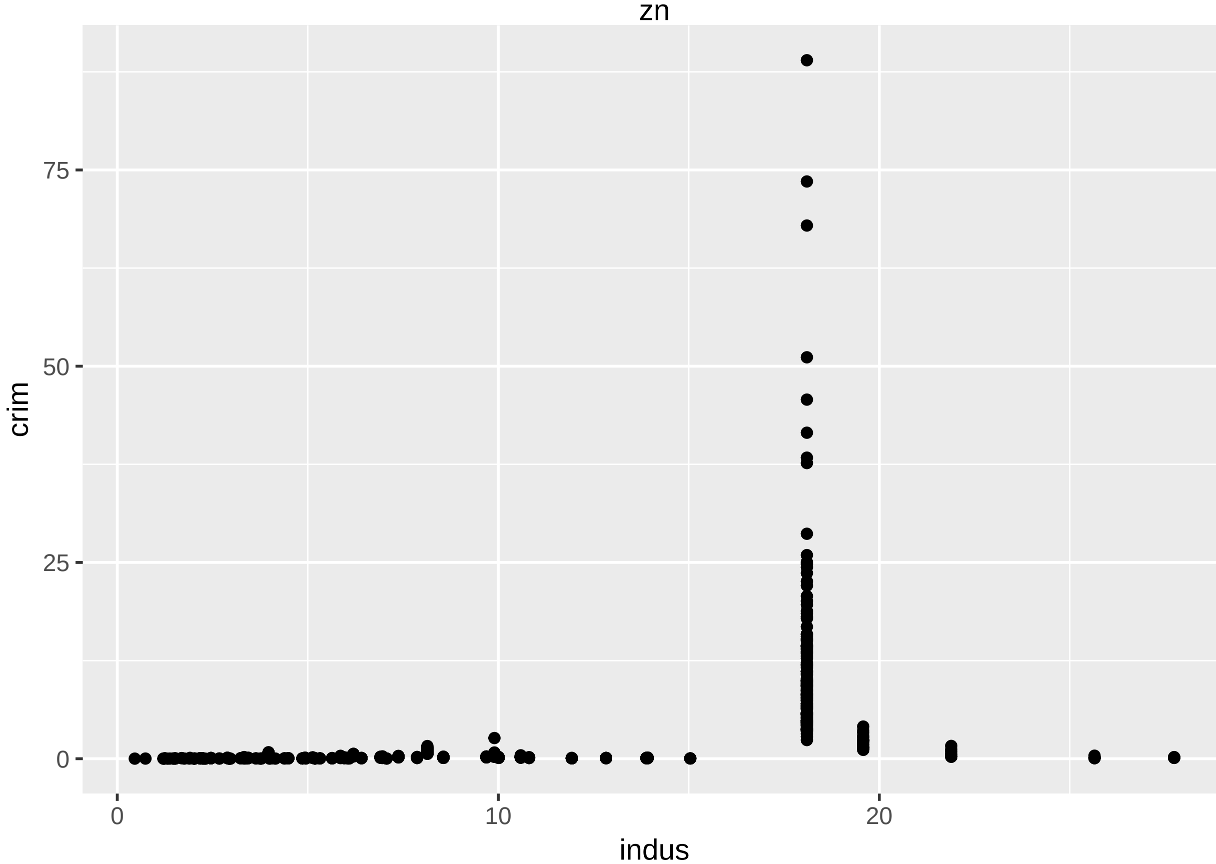
PartA

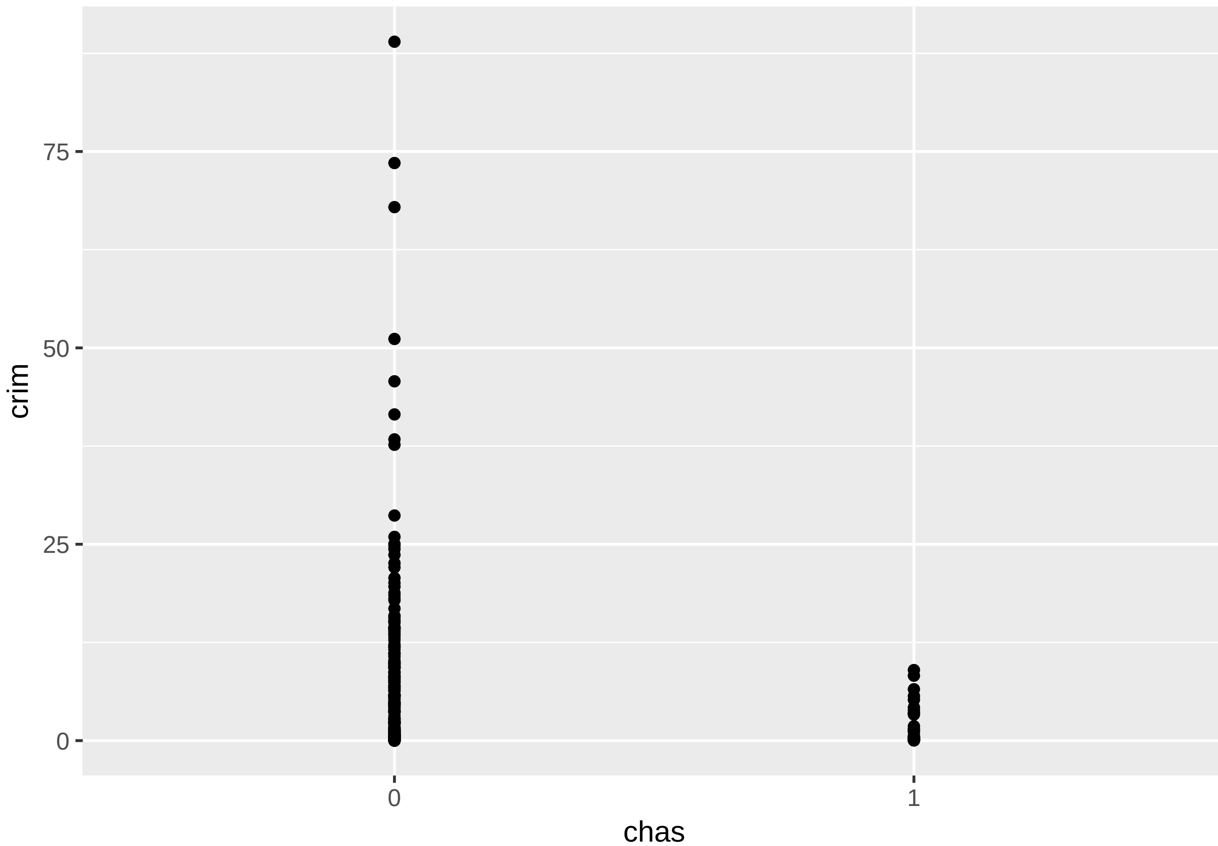
Question1

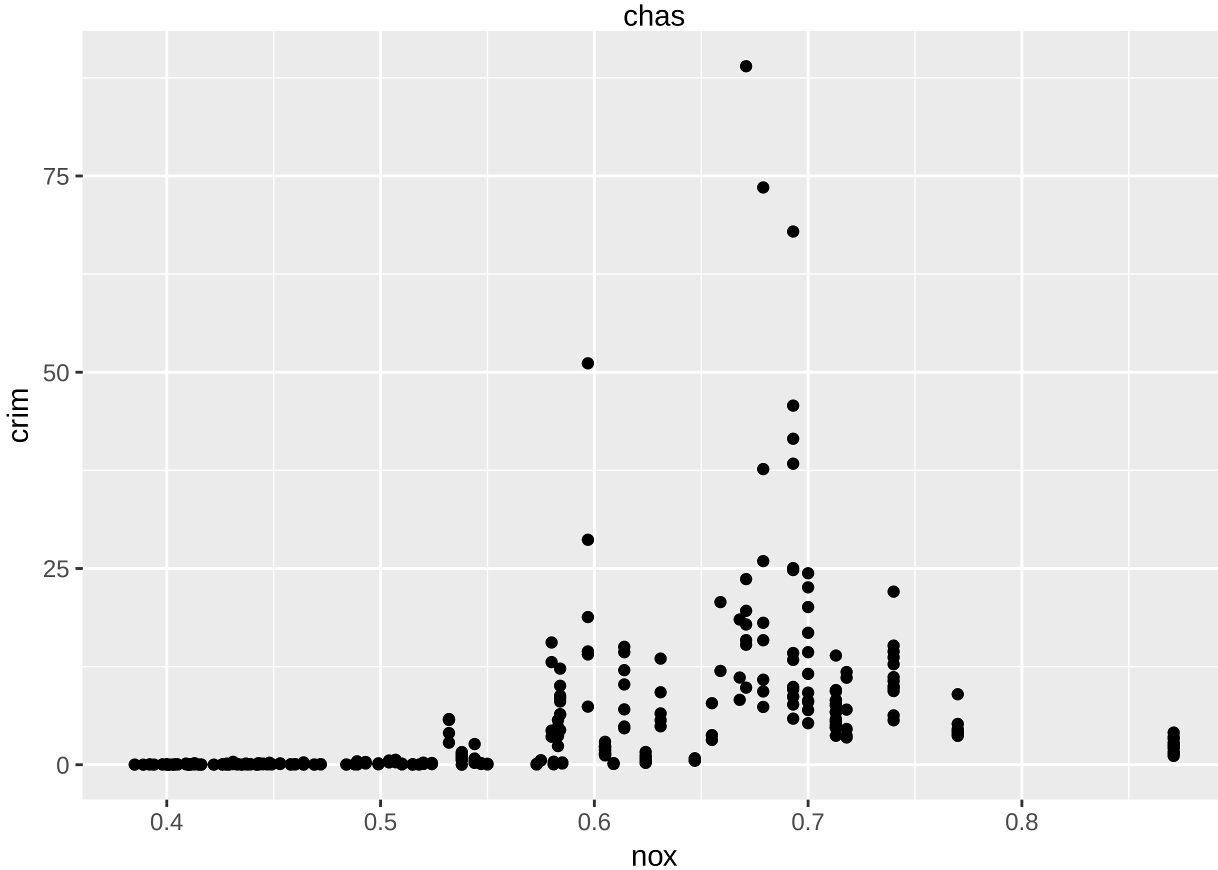
```
library(modelr)
library(ggplot2)
library(tidyverse)
library(mlbench)
data("BostonHousing")
data1=BostonHousing
for (i in names(data1)){
  print (data1%>%ggplot(aes_string(x=i,y="crim"))+geom_point())
}
```

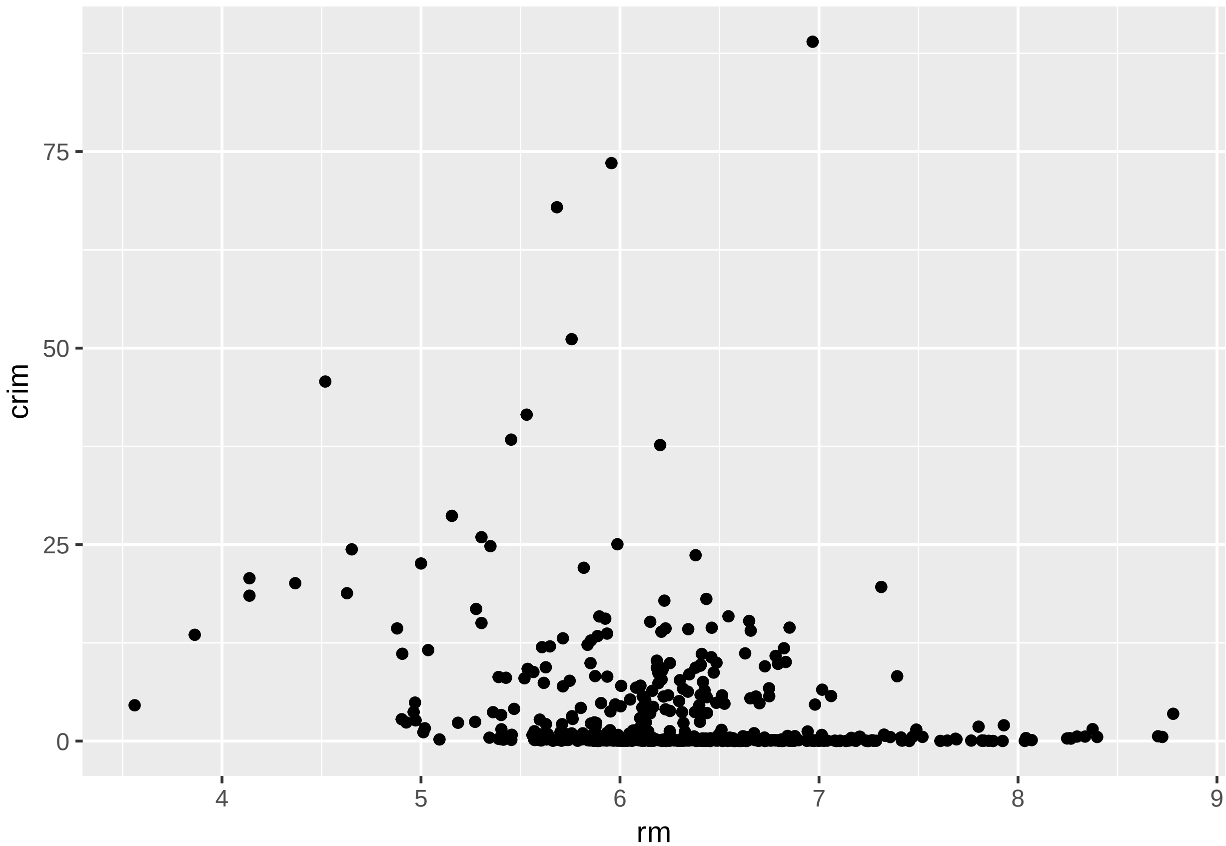



crim
0
25
50
75
0
25
50
75
100
zn

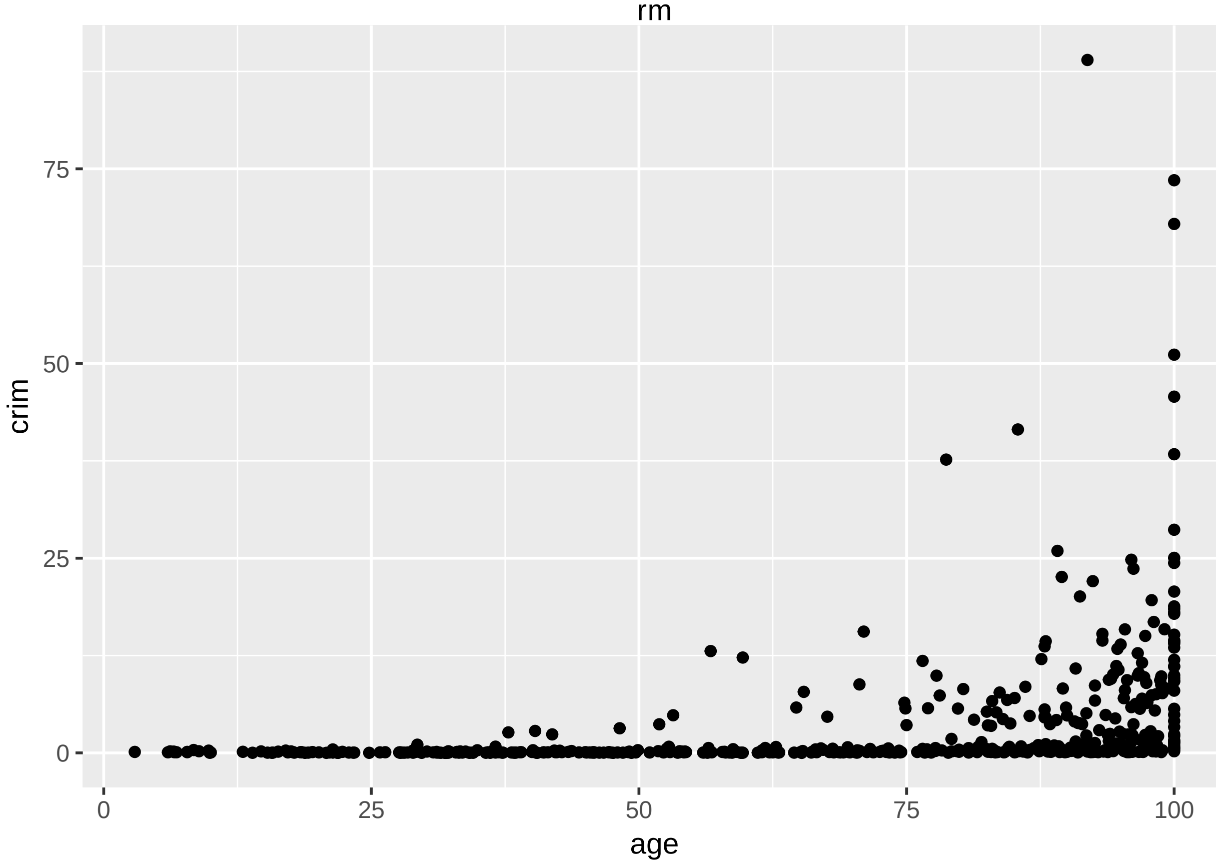

crim
0
25
50
75
0
10
20
indus


crim
75
50
25
0
0
1
chas

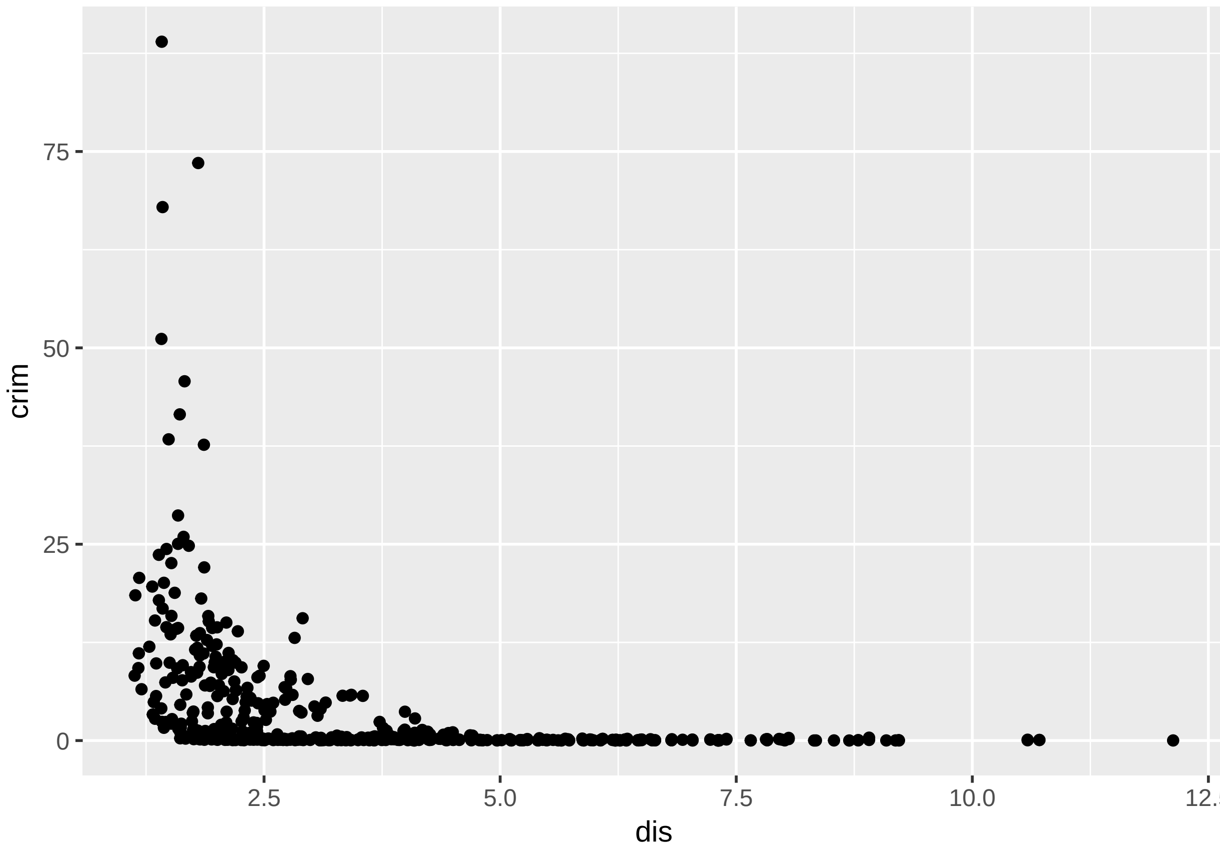

crim
75
50
25
0
0.4
0.5
0.6
0.7
0.8
nox



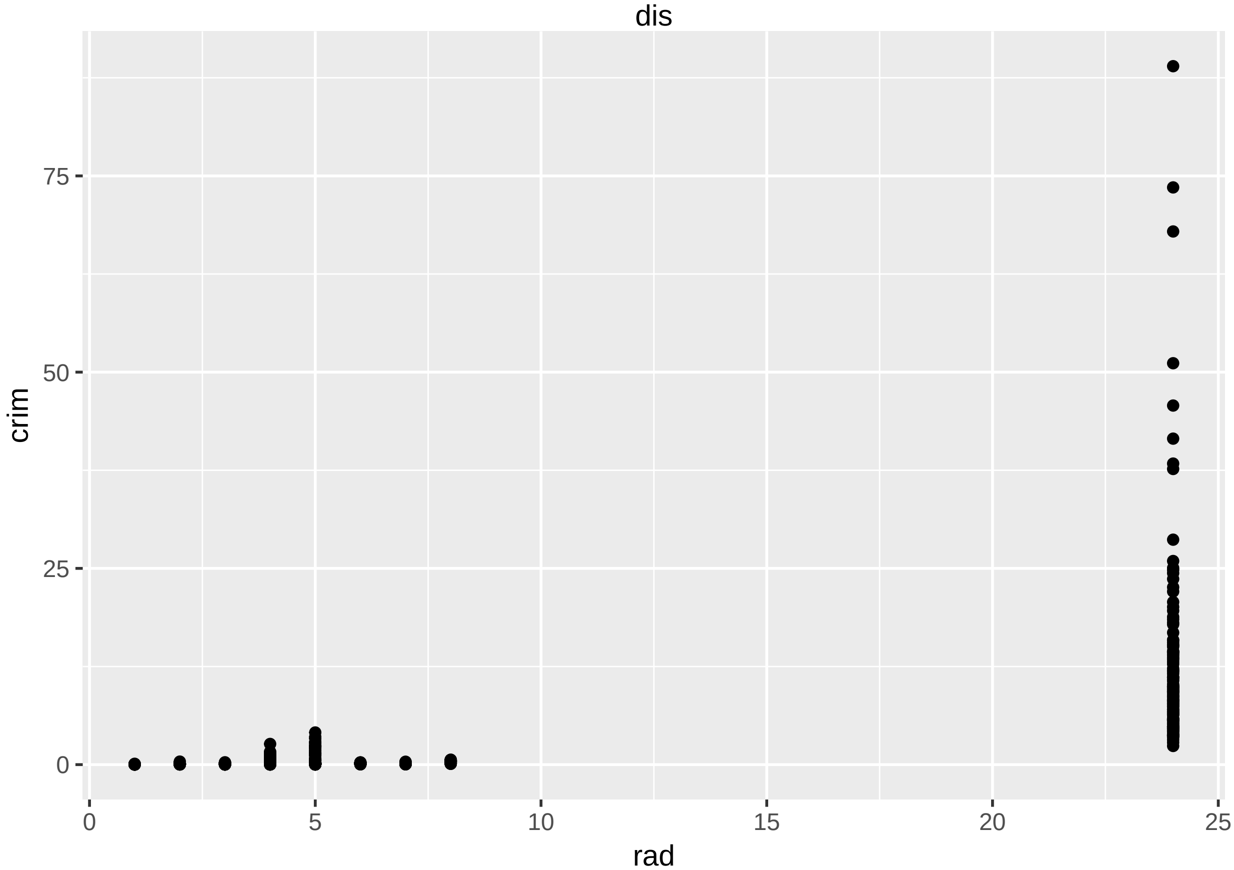
crim
0
25
50
75
4
5
6
7
8
9
rm



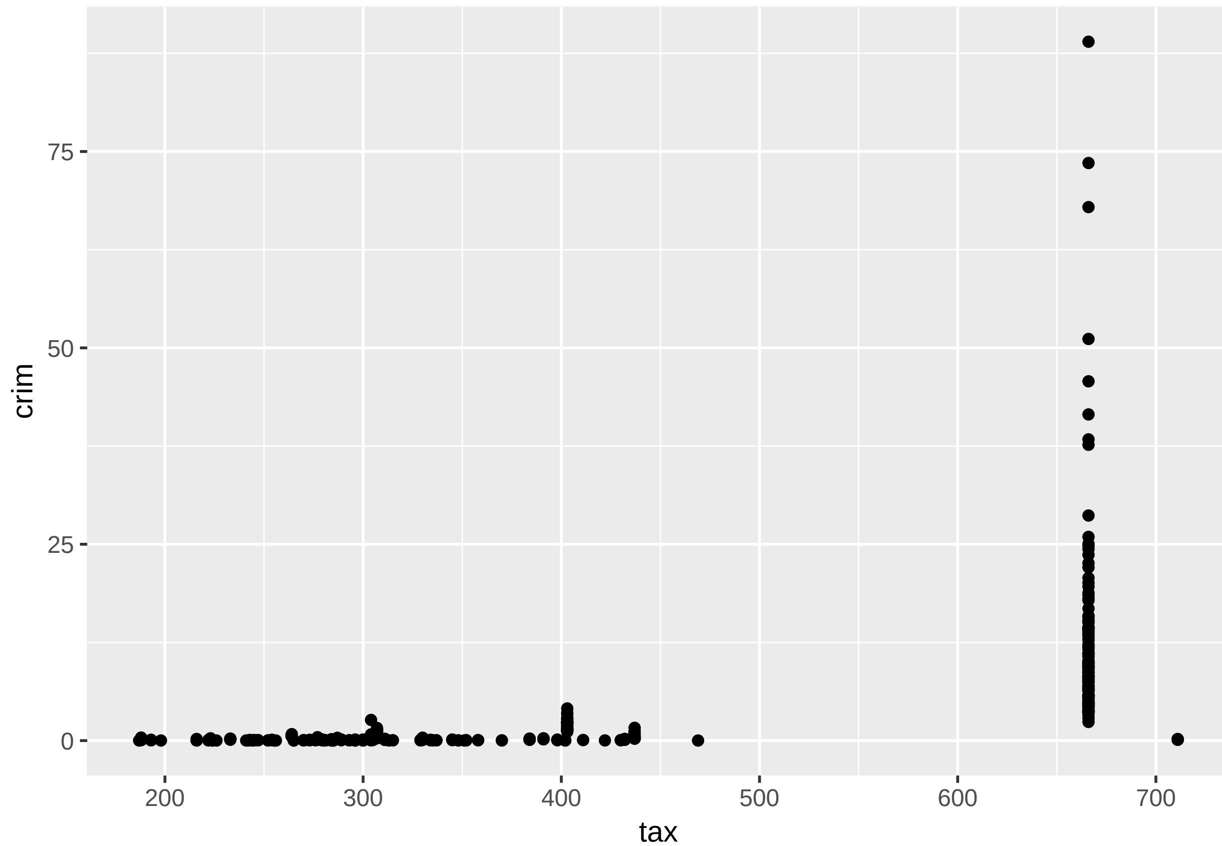
crim
0
25
50
75
0
25
50
75
100
age

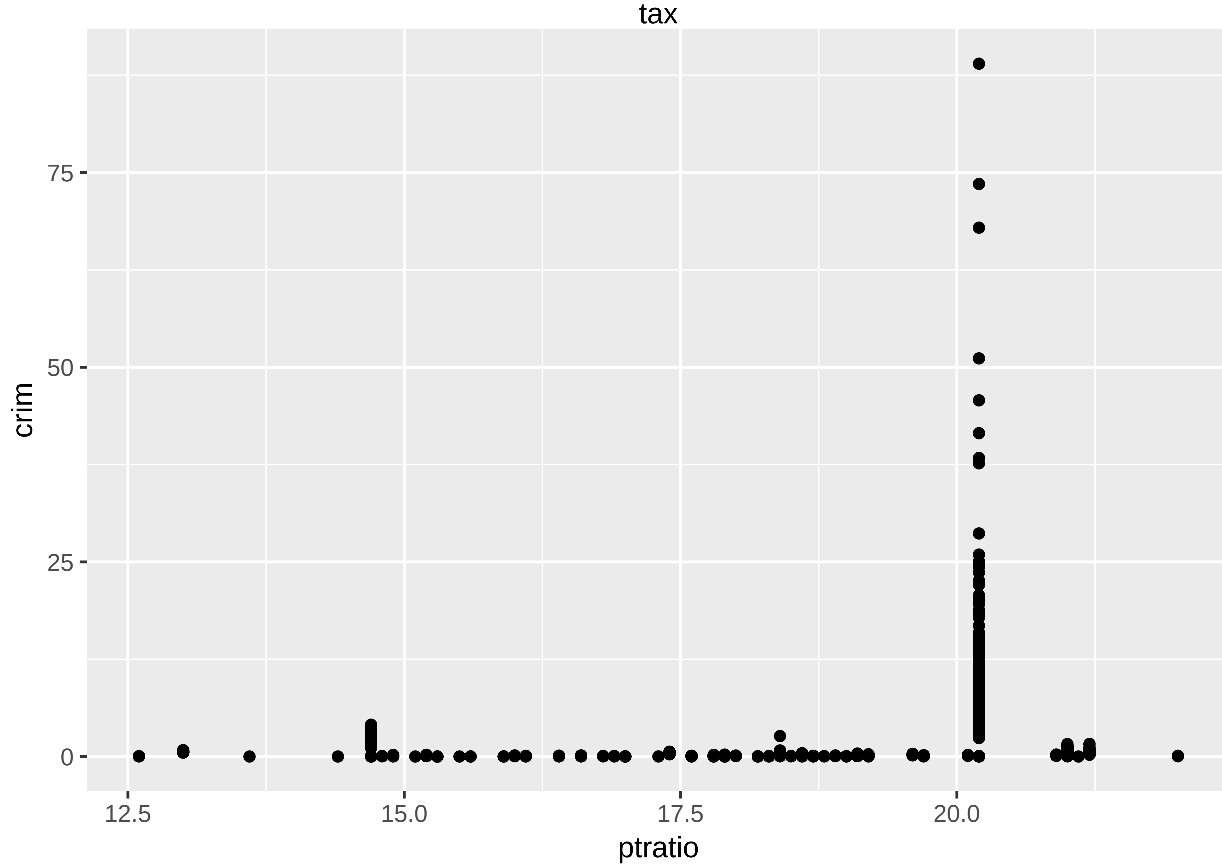


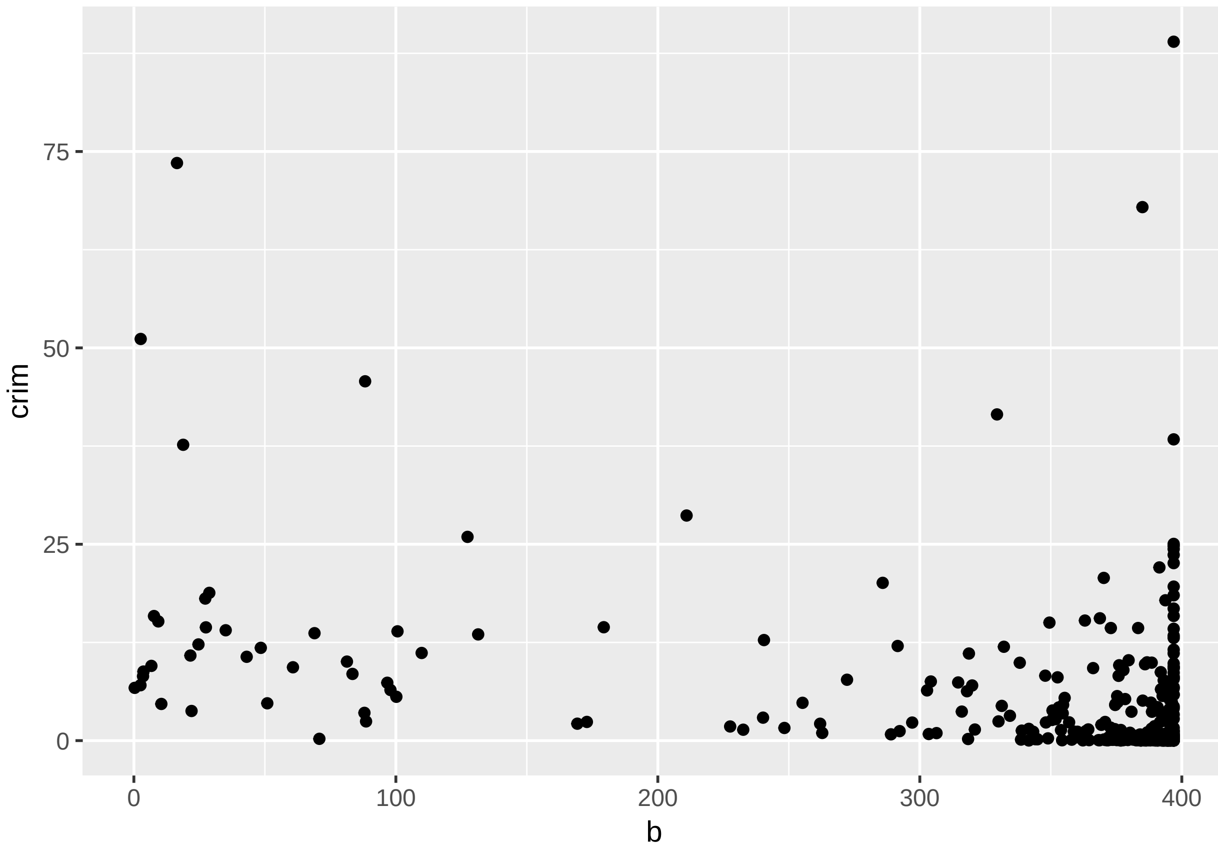
crim
75
50
25
0
2.5
5.0
7.5
10.0
12.5
dis

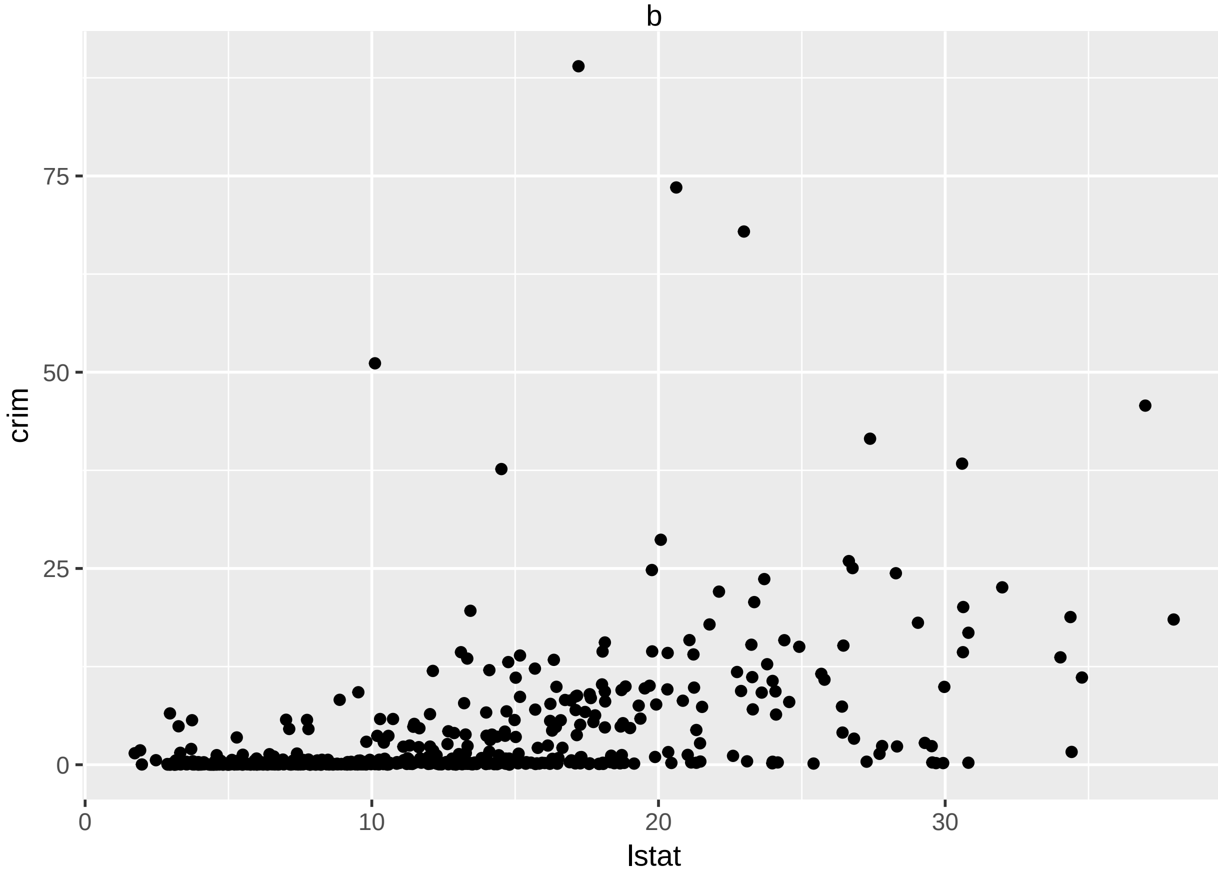


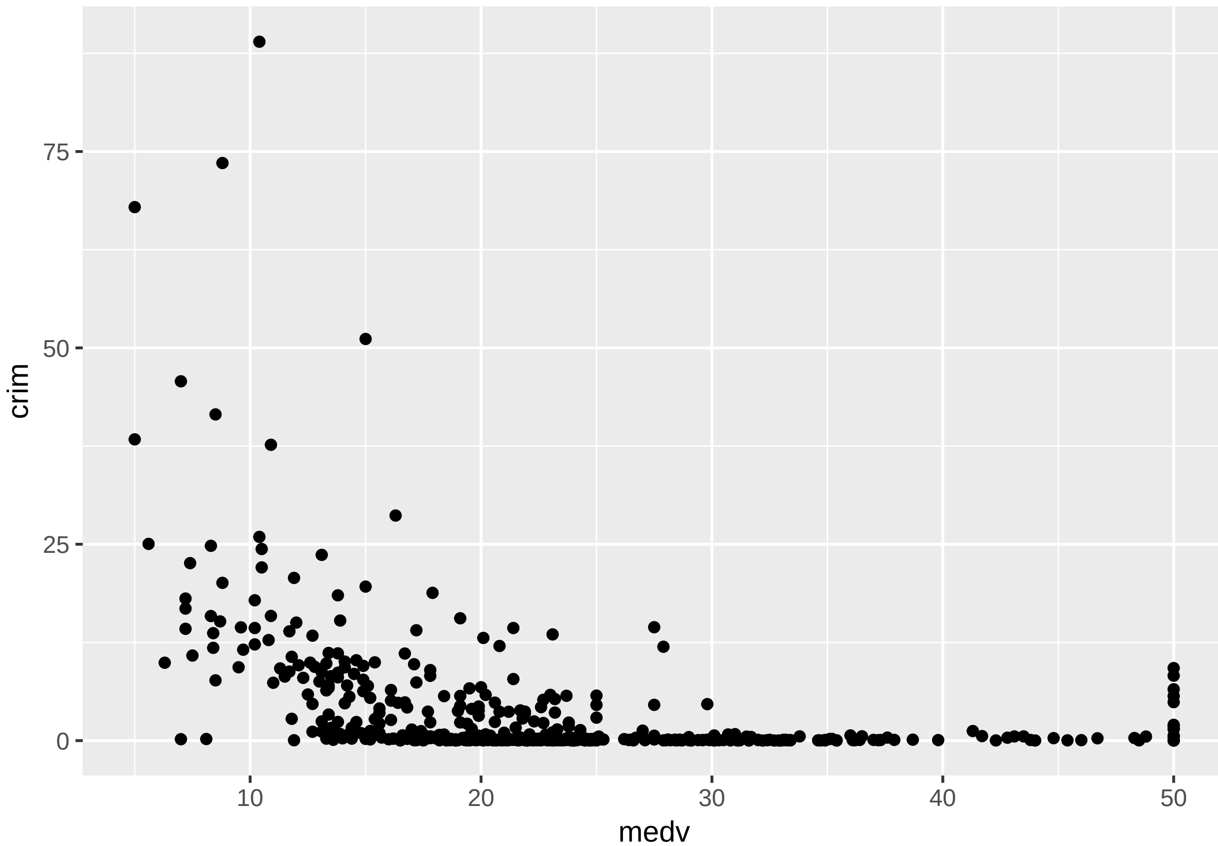
crim
75
50
25
0
0
5
10
15
20
25
rad


crim
75
50
25
0
200
300
400
500
600
700
tax

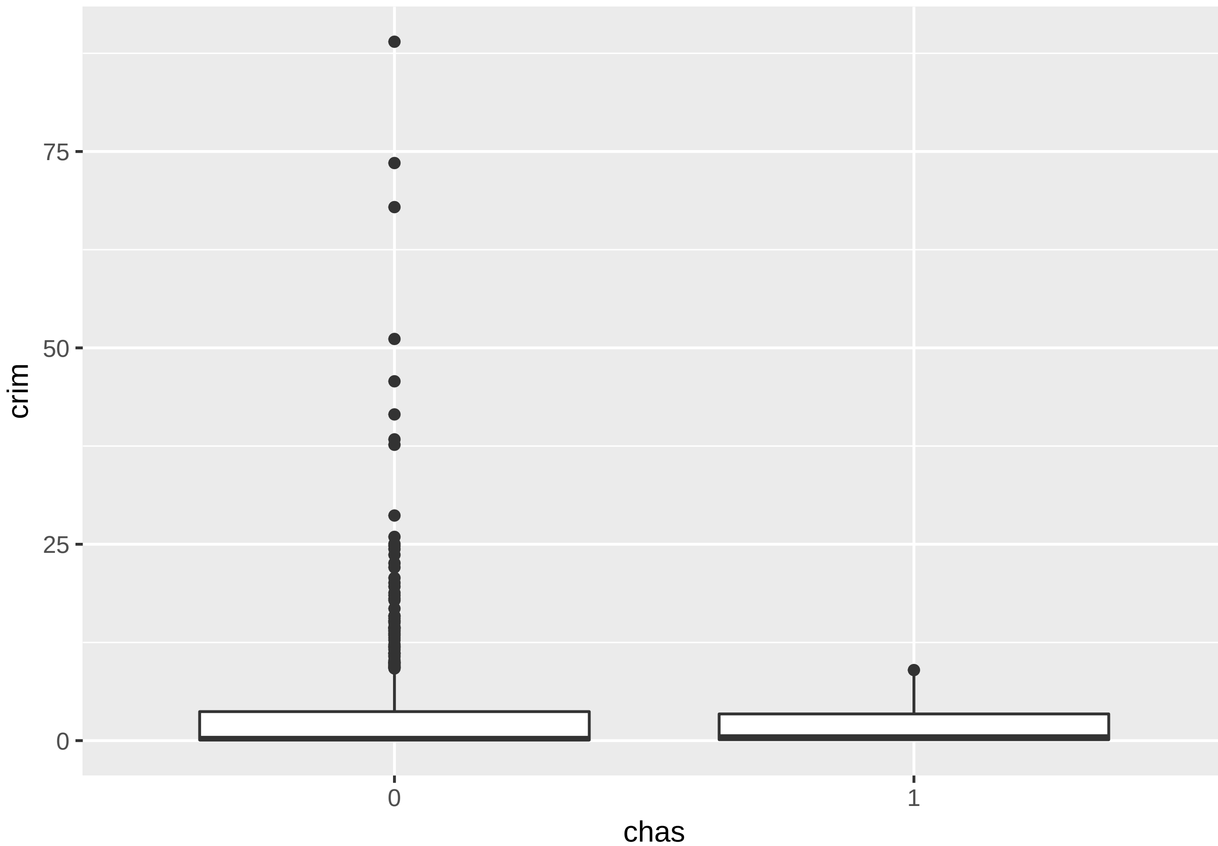

crim
75
50
25
0
12.5
15.0
17.5
20.0
ptratio


crim
75
50
25
0
0
100
200
300
400
b

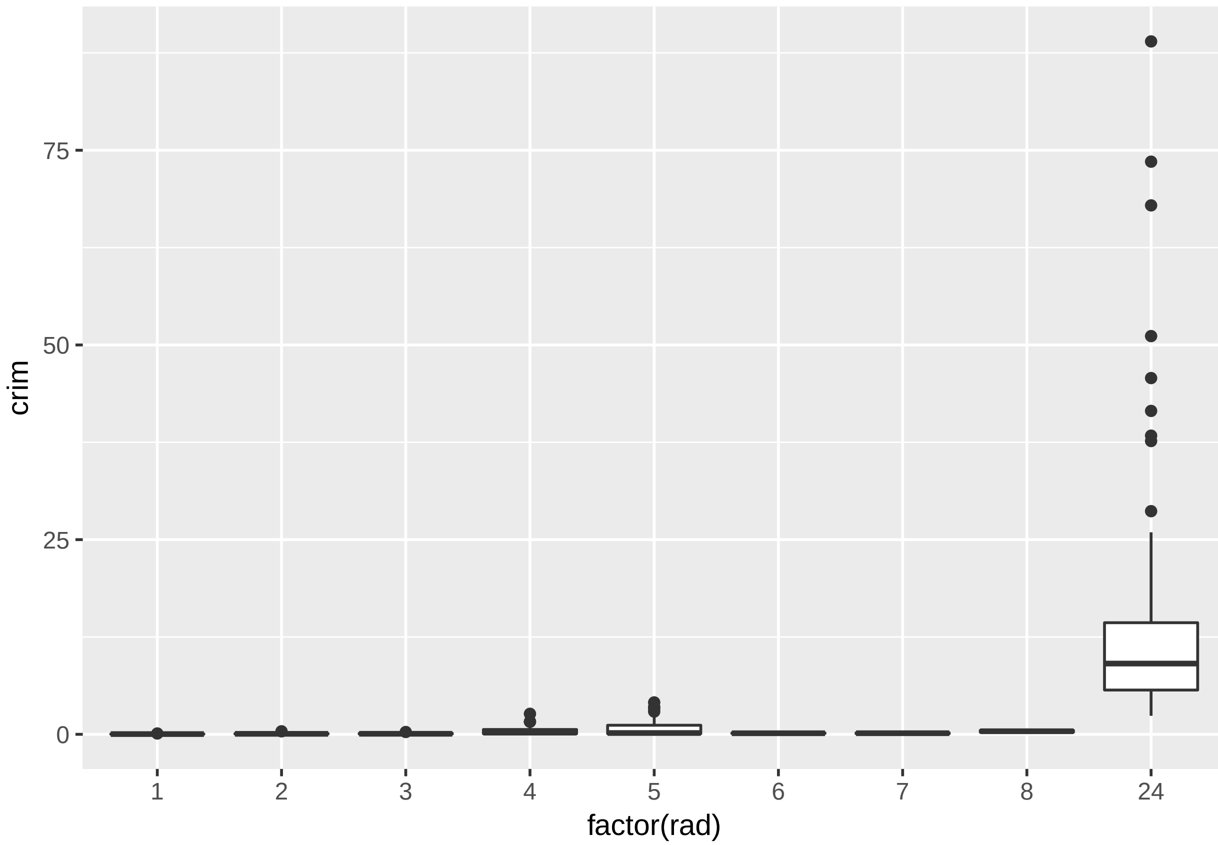

crim
75
50
25
0
0
10
20
30
lstat




```
#Categorical variables, how to make them looks like a boxplot
data1%>%ggplot(aes(x=chas,y=crim))+geom_boxplot()
```

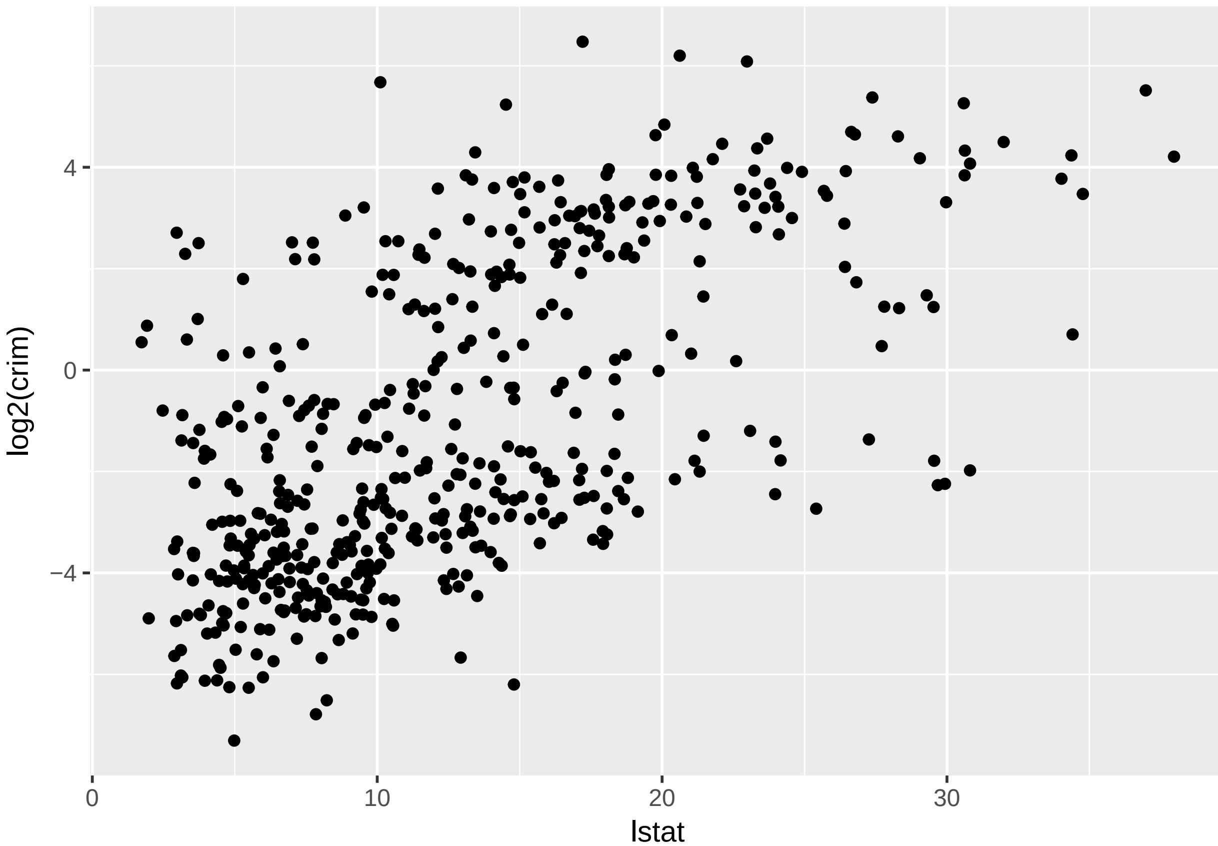



```
data1%>%ggplot(aes(x=factor(rad),y=crim))+geom_boxplot()
```

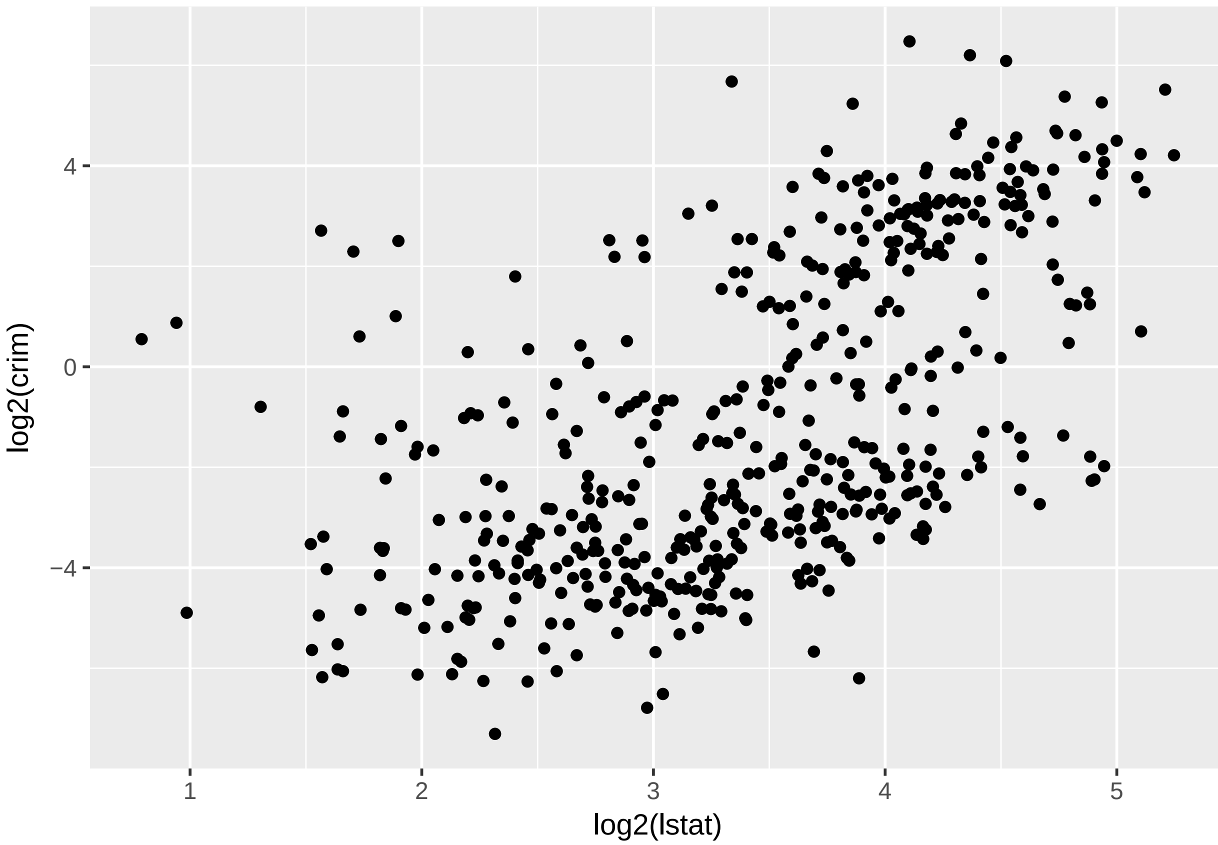


I found that Lstat and dis are the most predictive. So next thing is to do some transformations over those two parameters in order to figure out the best linear relationship.

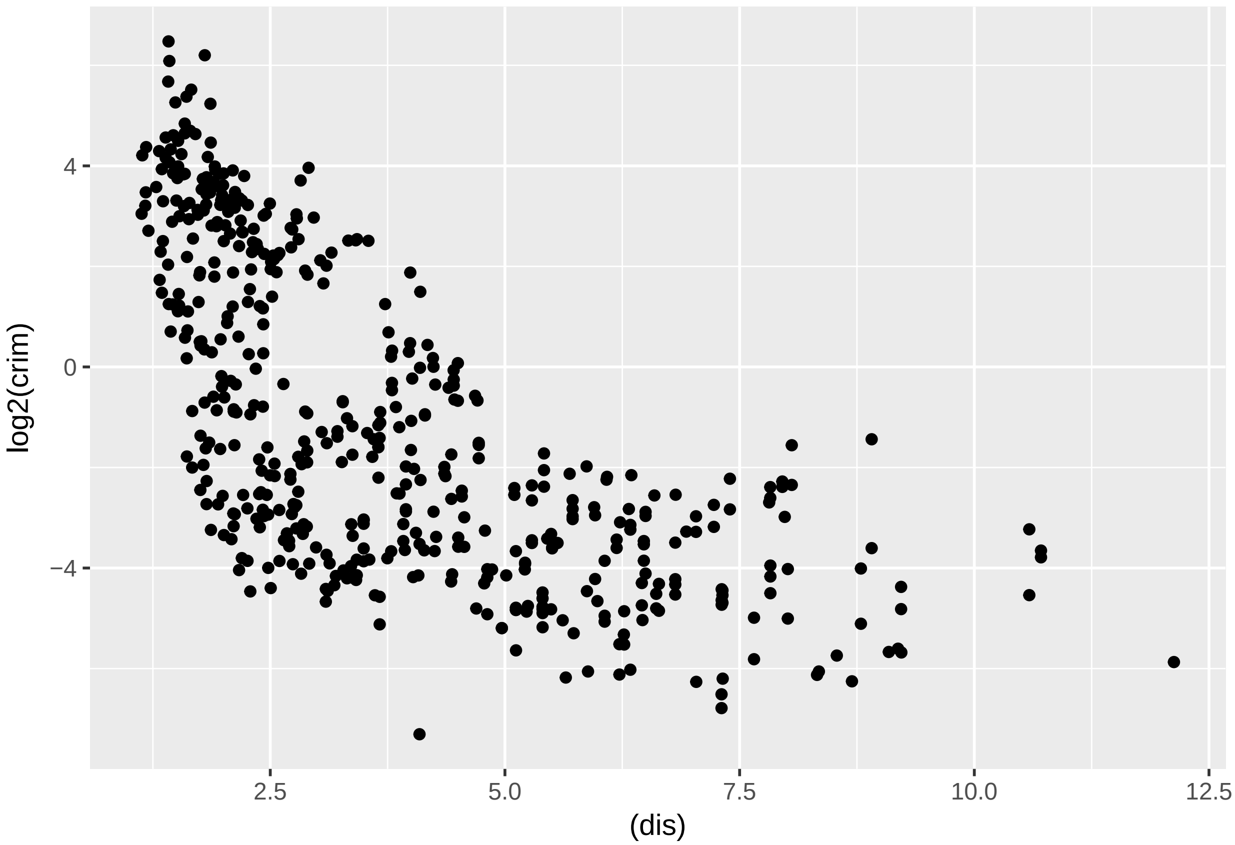
```
data1%>%ggplot(aes(x=lstat,y=log2(crim)))+geom_point()
```



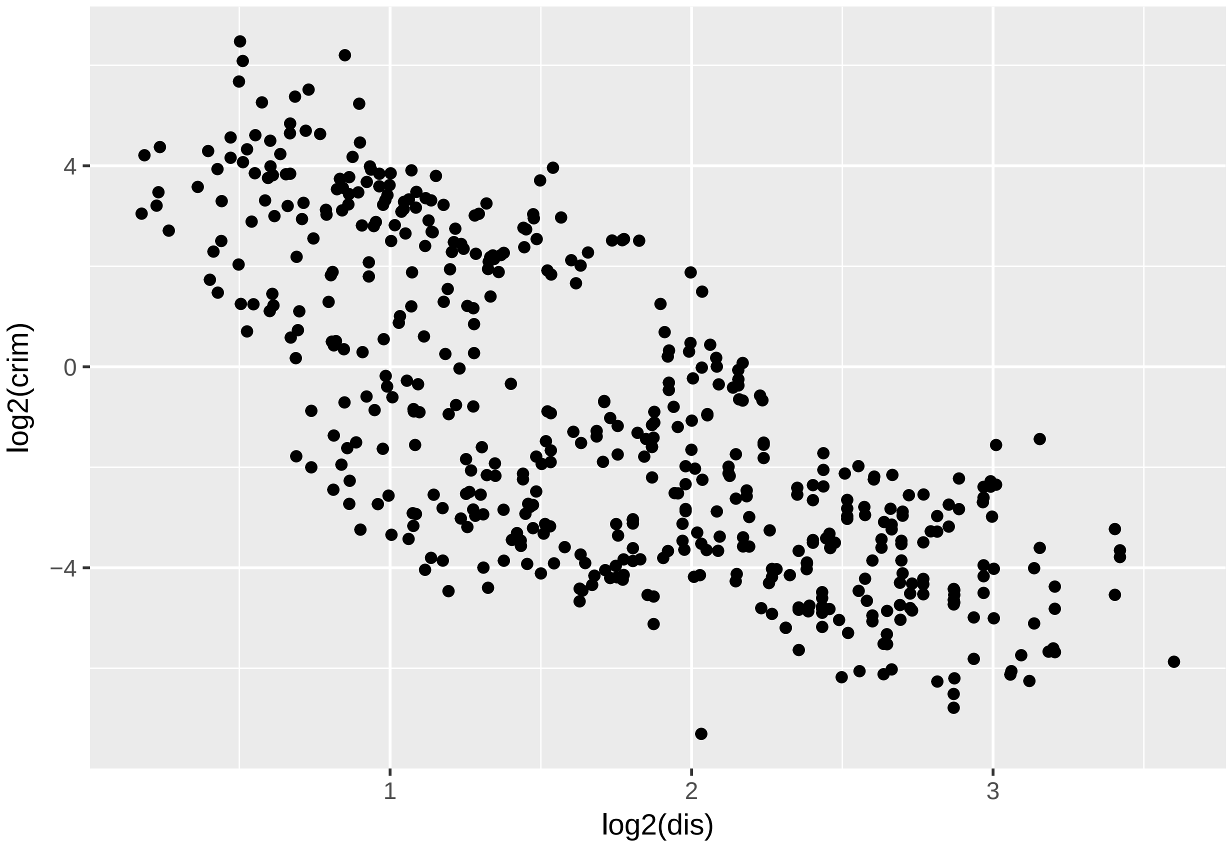

```
data1%>%ggplot(aes(x=log2(lstat),y=log2(crim)))+geom_point()
```




```
data1%>%ggplot(aes(x=(dis),y=log2(crim)))+geom_point()
```




```r
data1%>%ggplot(aes(x=log2(dis),y=log2(crim)))+geom_point()
```



I found the plot where x=log2(dis), y=log2(crim) has a strong linear relationship. Therefore, I decided to predict log2(crim) using log2(dis)

```
fit1 <- lm(log2(crim)~log2(dis),data=data1)
summary(fit1)
```

```
##
## Call:
## lm(formula = log2(crim) ~ log2(dis), data = data1)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -5.232 -1.608 -0.027  1.800  4.751
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   3.9835     0.2245   17.74   <2e-16 ***
## log2(dis)    -2.9810     0.1193  -24.99   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.086 on 504 degrees of freedom
## Multiple R-squared:  0.5534, Adjusted R-squared:  0.5525
## F-statistic: 624.6 on 1 and 504 DF,  p-value: < 2.2e-16
```

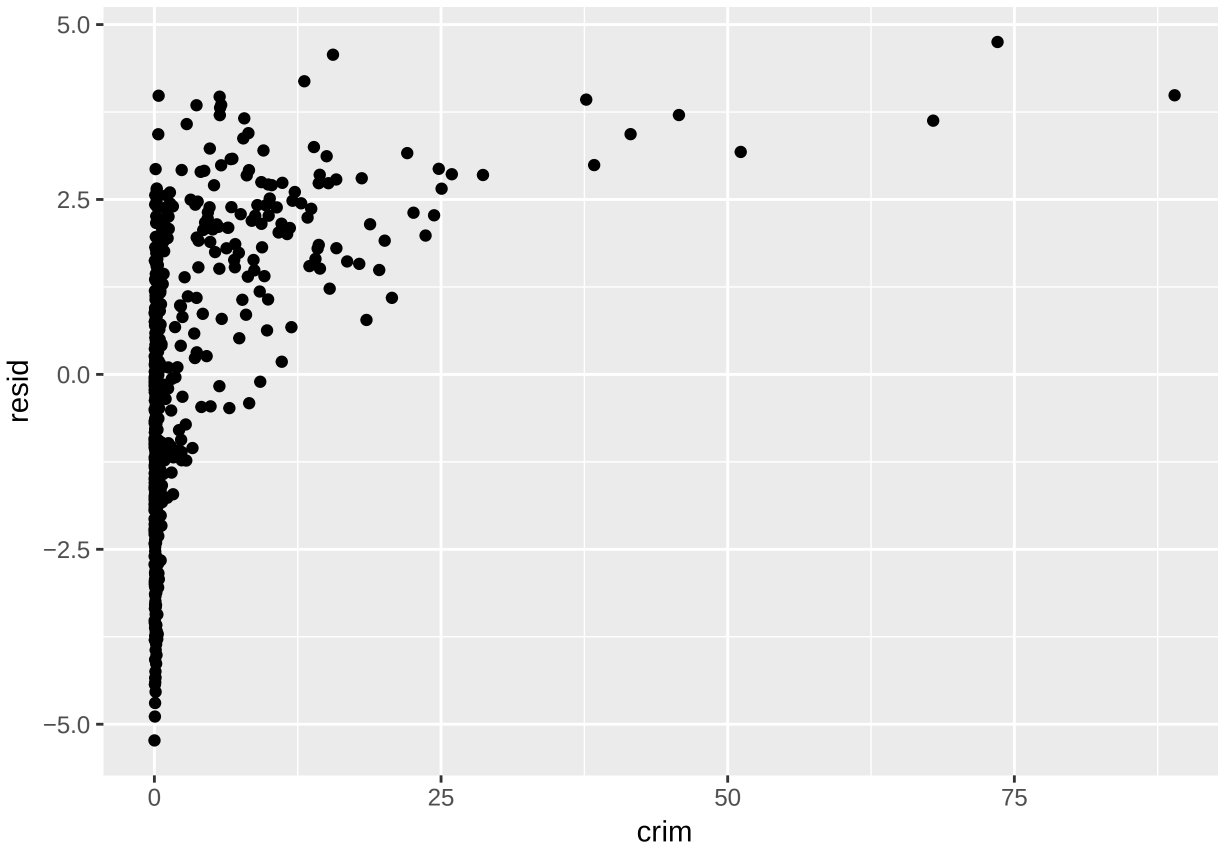
```
coef(fit1)
```

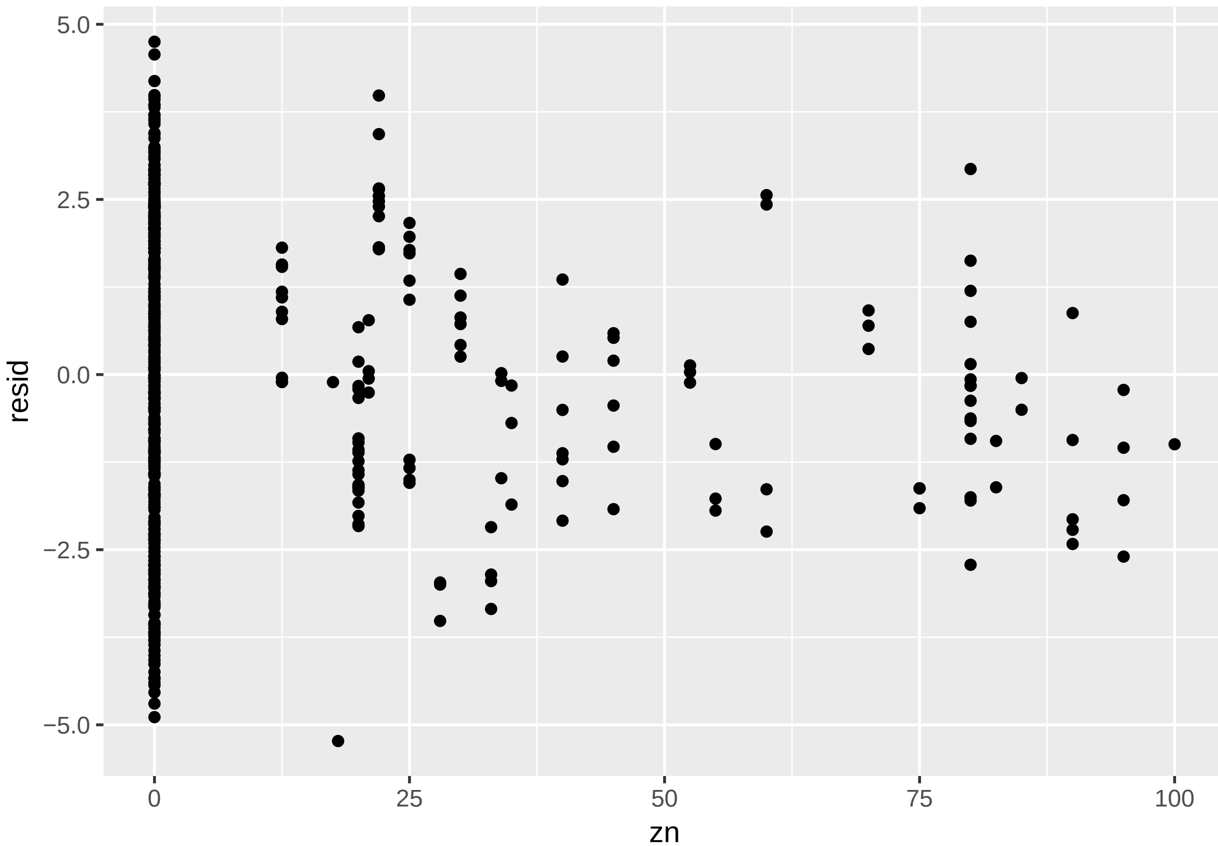
```
## (Intercept)   log2(dis)
##     3.98346    -2.98103
```

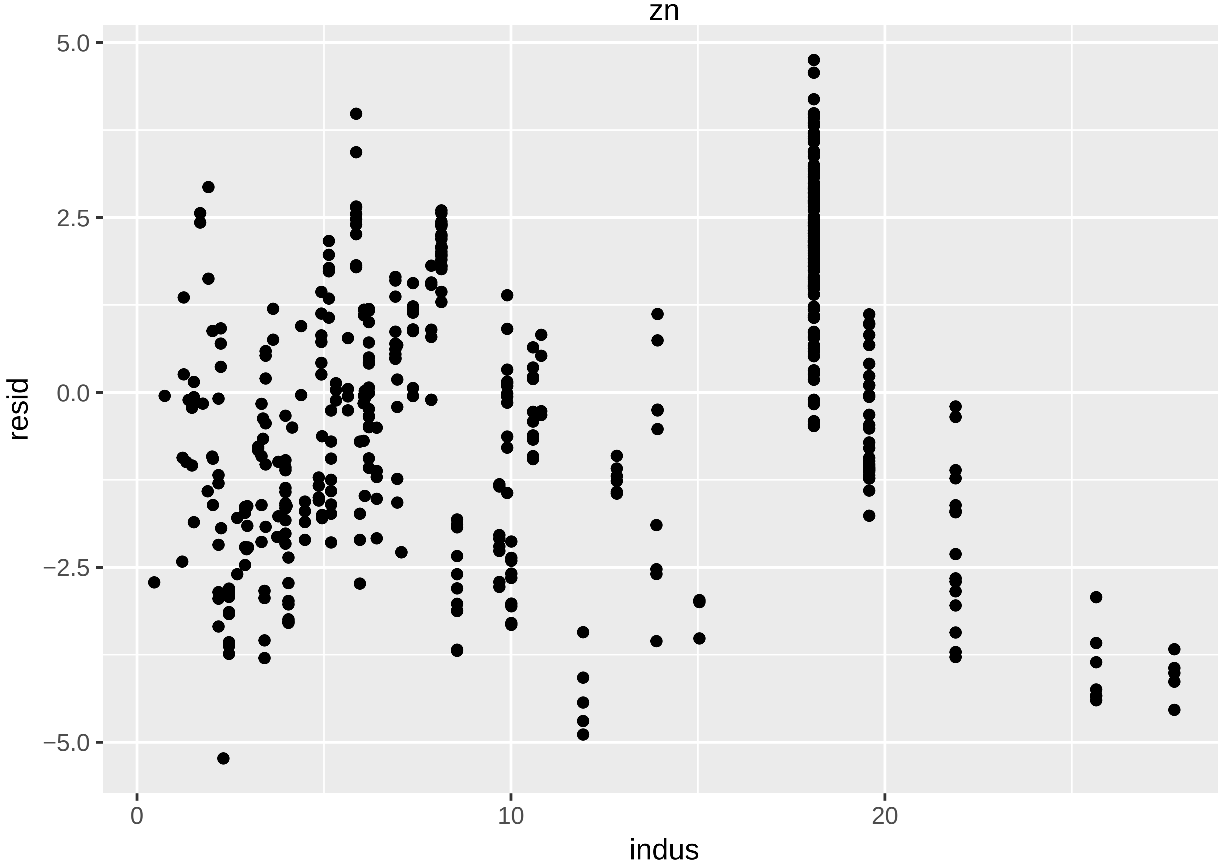
The above is the summary of the fitted model parameters.

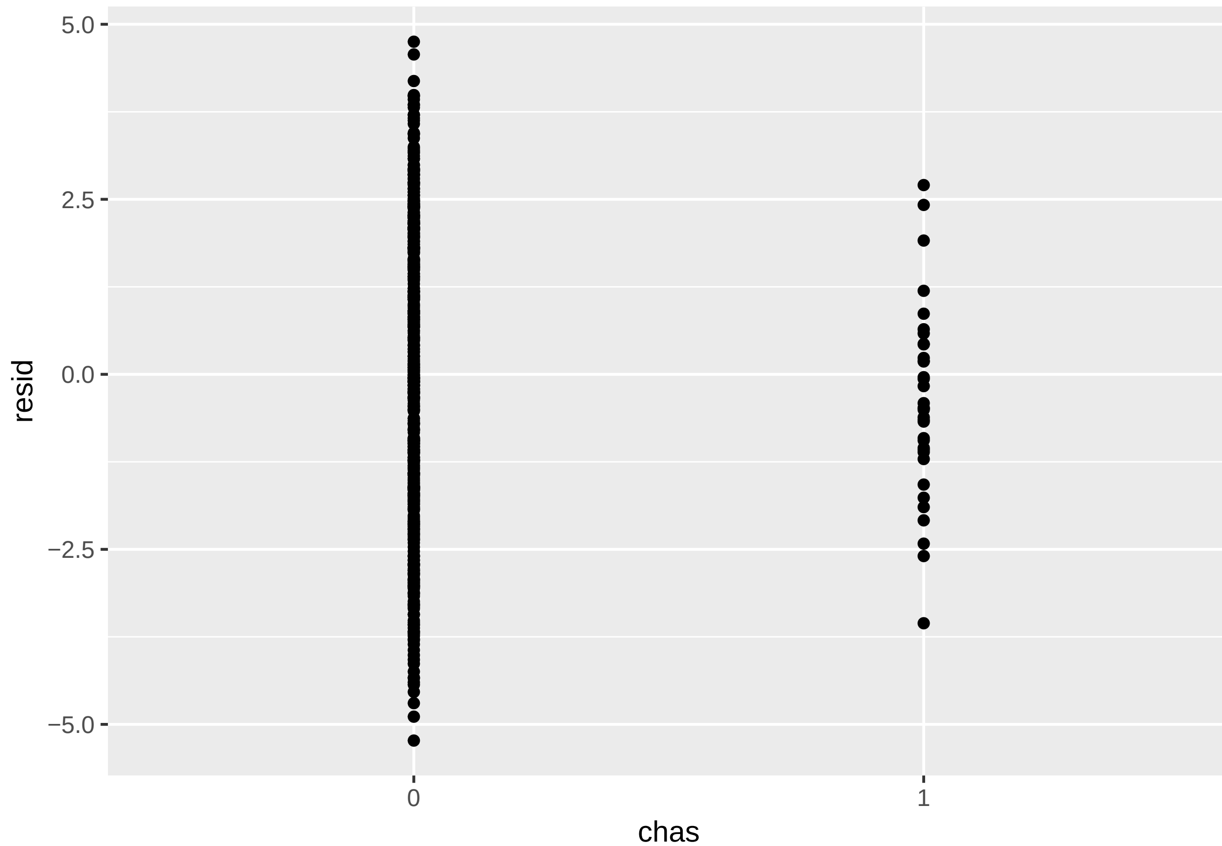
Problem 2

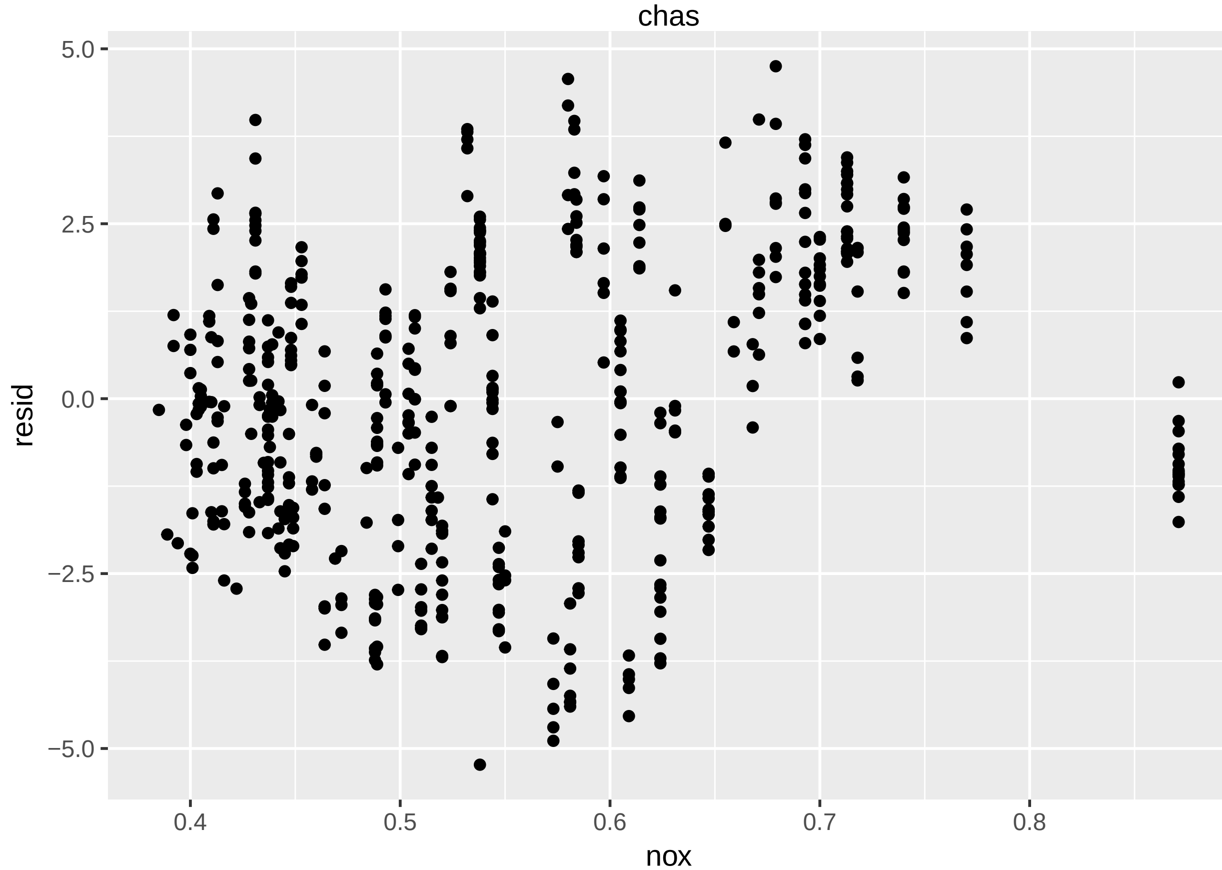
```
for (i in names(data1)){
  print (data1%>% add_residuals(fit1)%>%
           ggplot()+geom_point(aes_string(x=i,y="resid")))
}
```

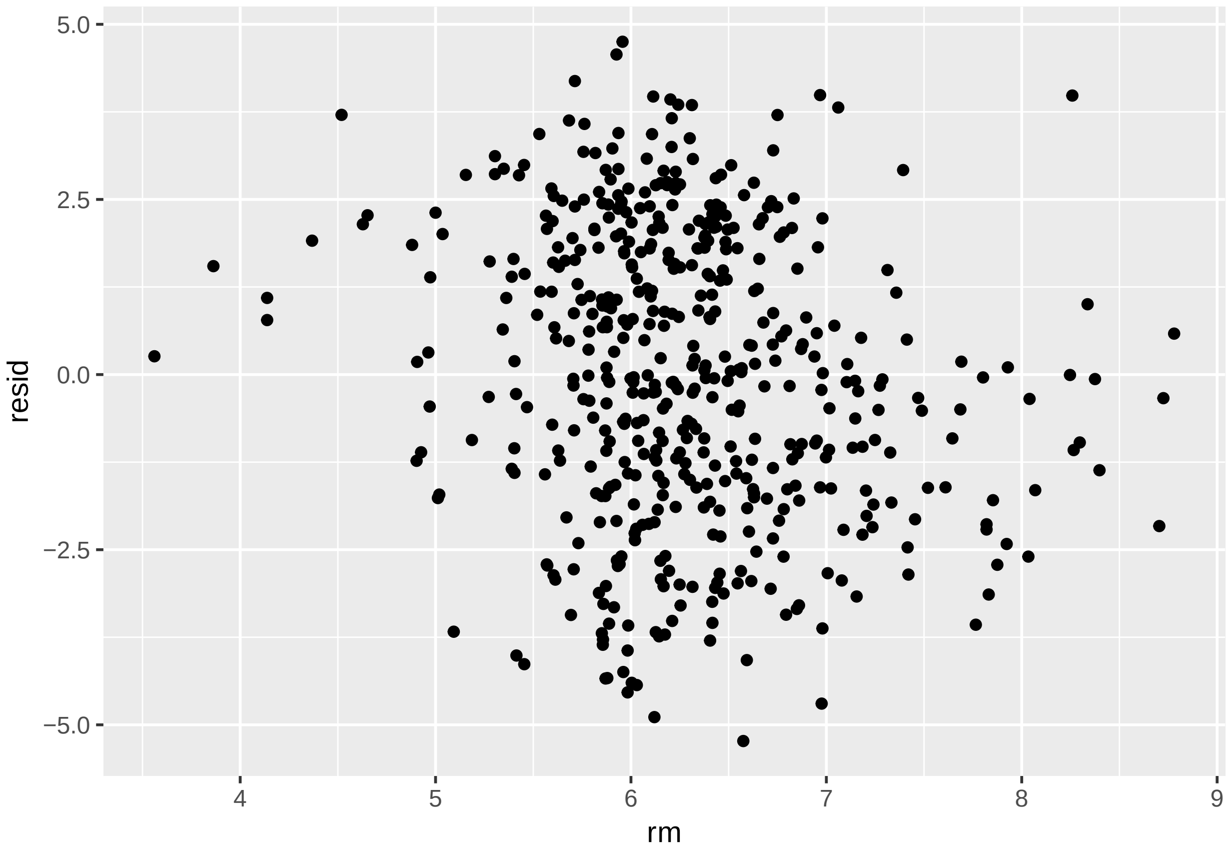


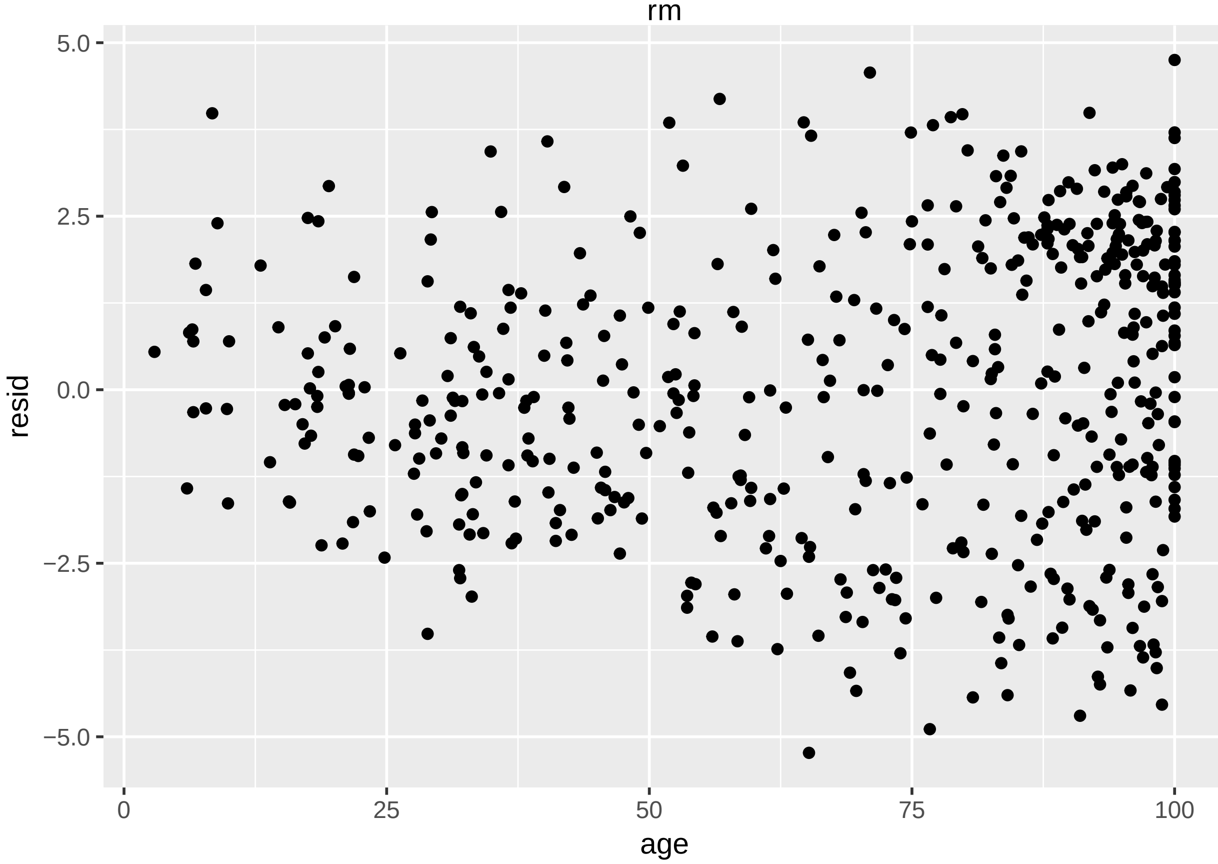

5.0
2.5
0.0
−2.5
−5.0
resid
0
25
50
75
100
zn

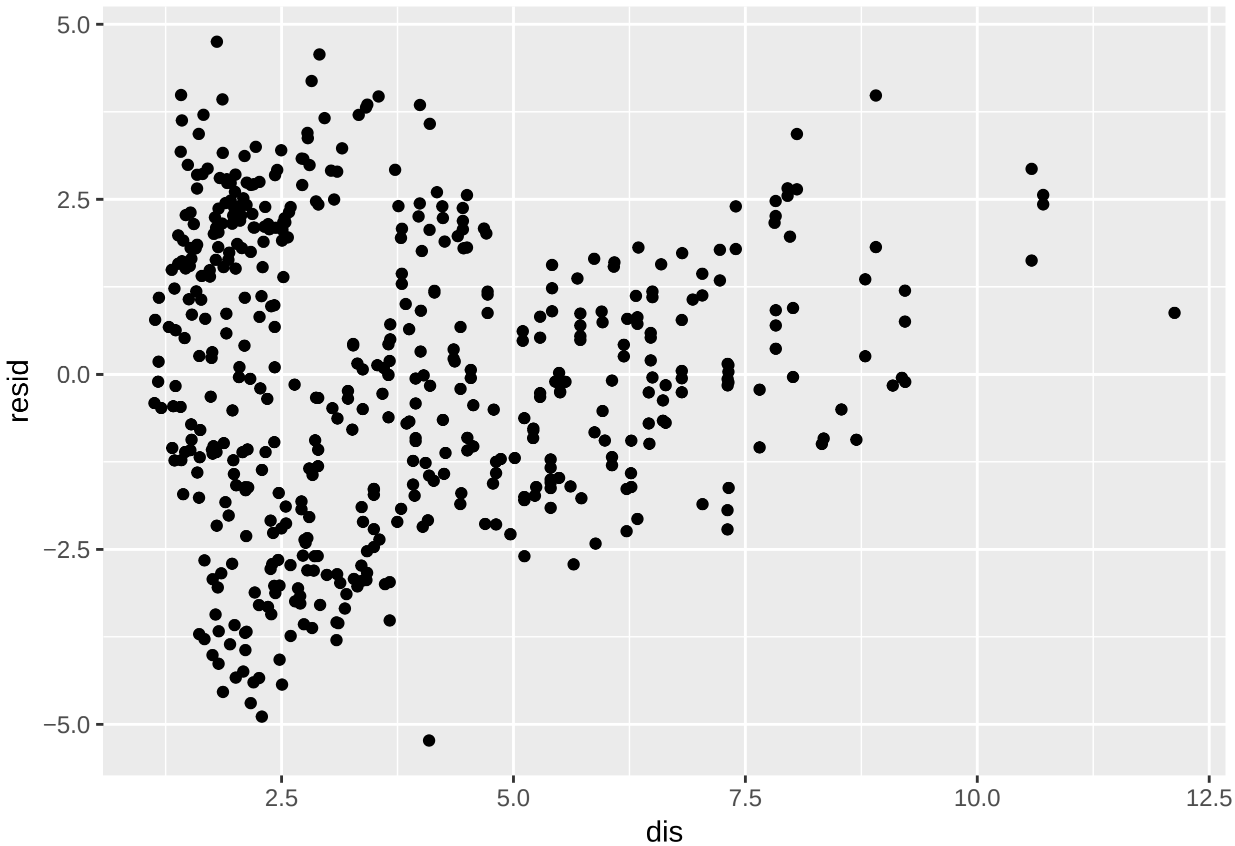

5.0
2.5
0.0
−2.5
−5.0
resid
0
10
20
indus


5.0
2.5
0.0
−2.5
−5.0
resid
0
1
chas

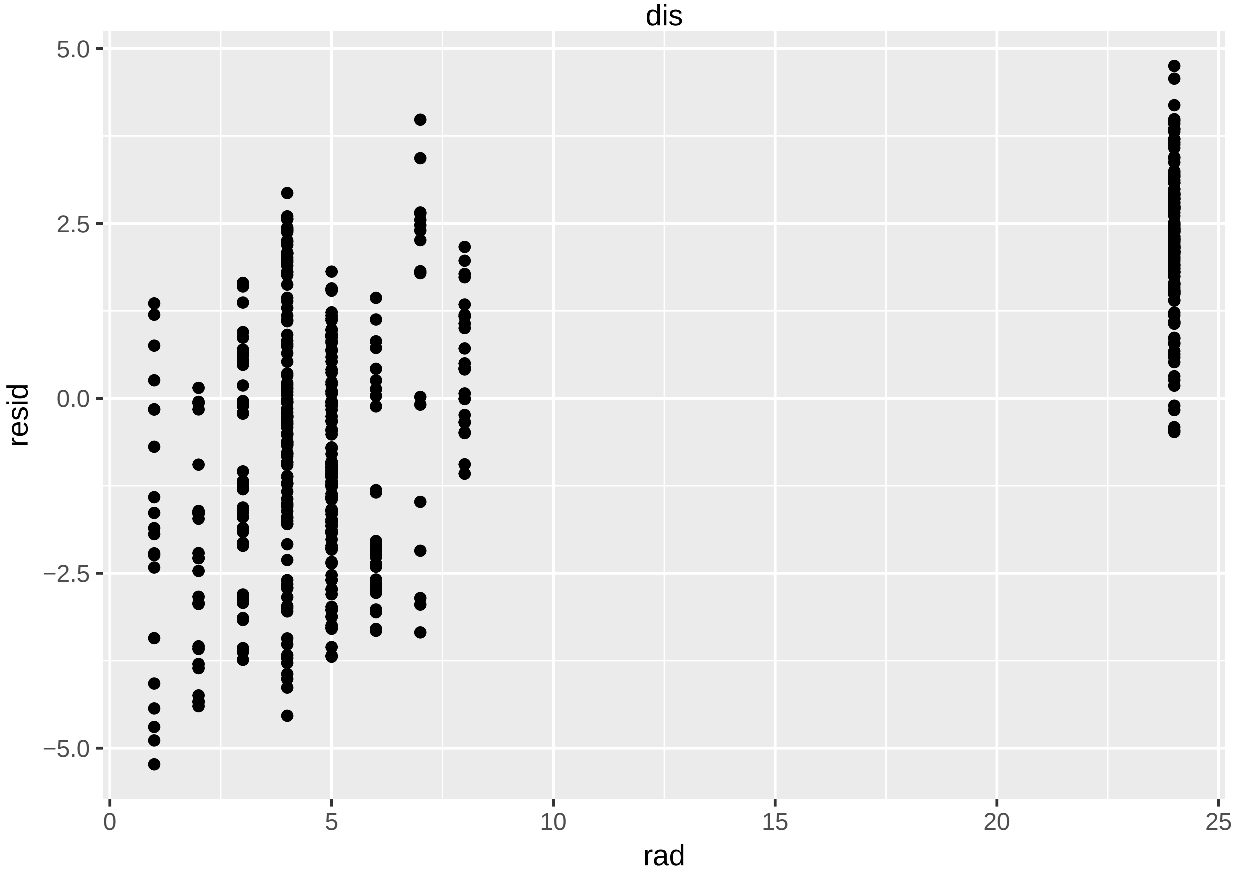

5.0
2.5
0.0
−2.5
−5.0
resid
0.4
0.5
0.6
0.7
0.8
nox


5.0
2.5
0.0
−2.5
−5.0
resid
4
5
6
7
8
9
rm

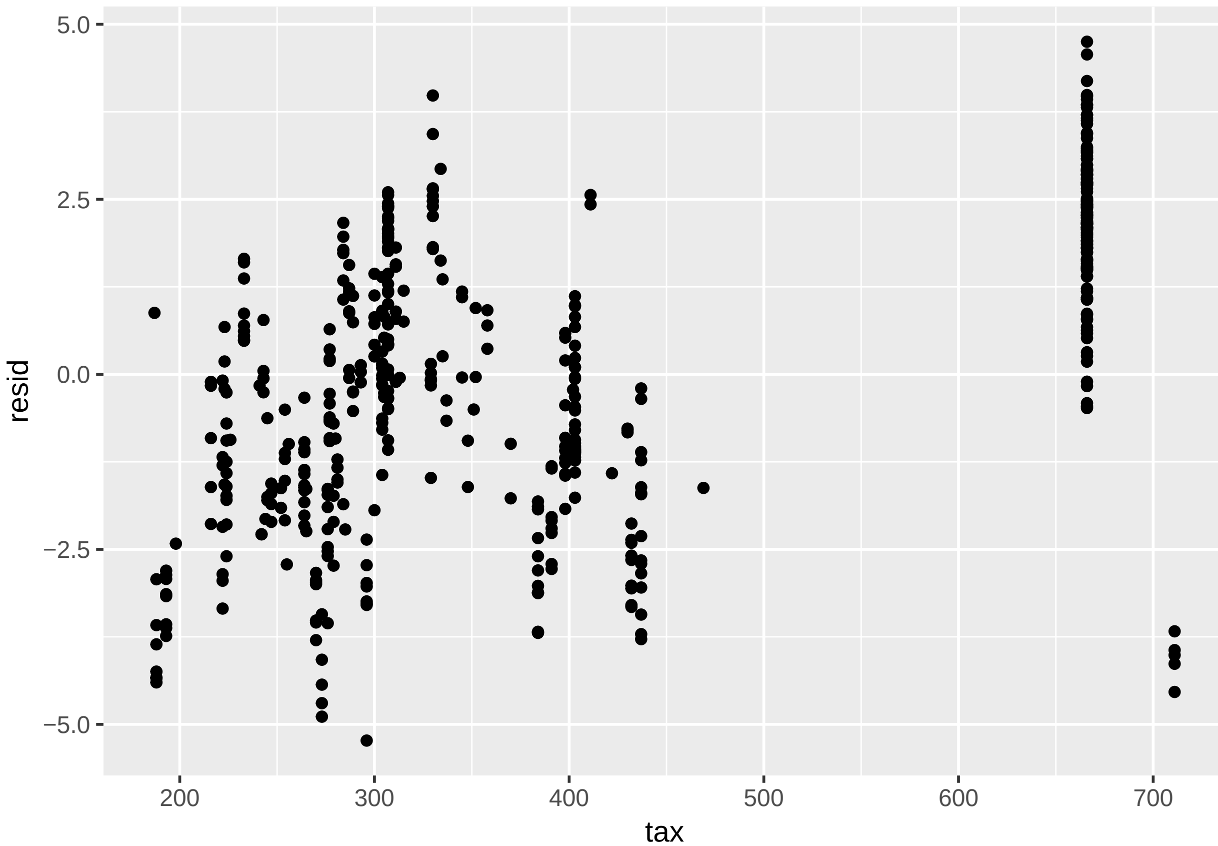

5.0
2.5
0.0
−2.5
−5.0
resid
0
25
50
75
100
age

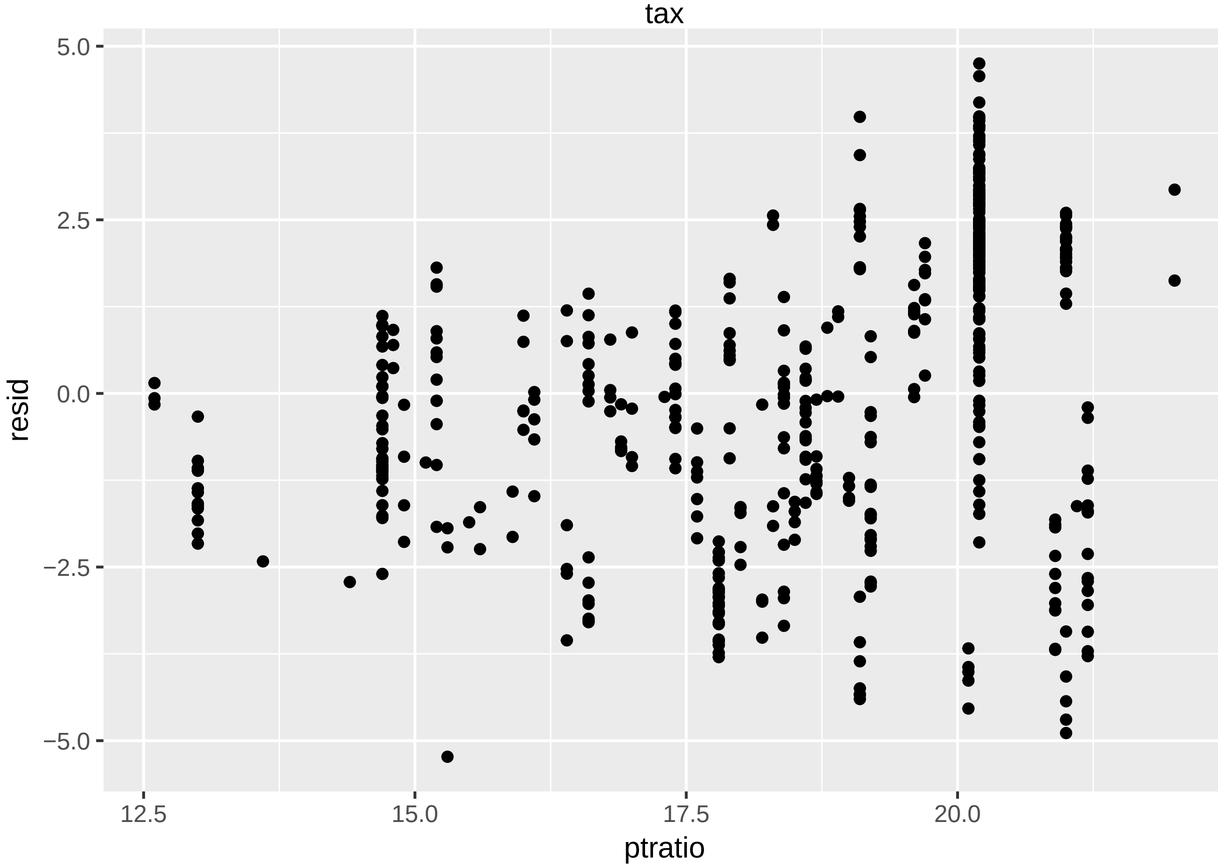


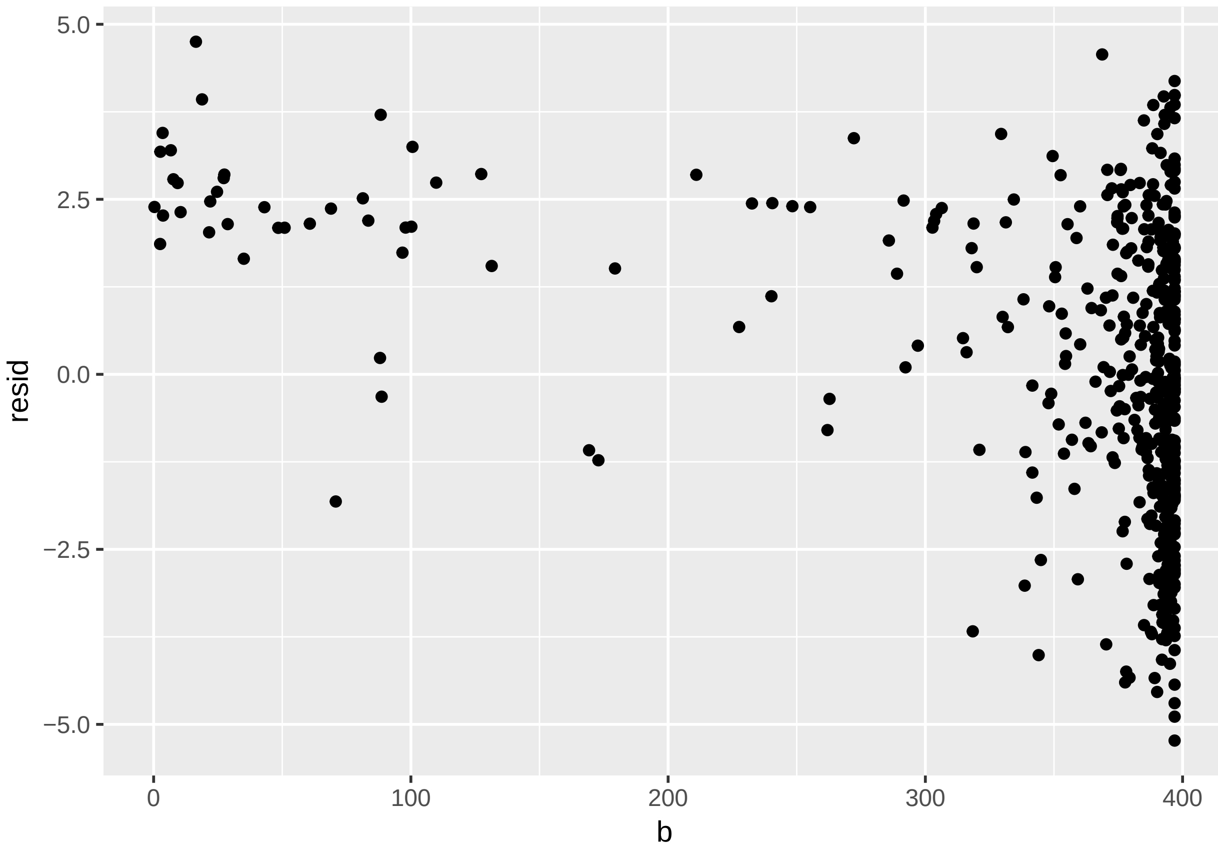
5.0
2.5
0.0
−2.5
−5.0
resid
2.5
5.0
7.5
10.0
12.5
dis



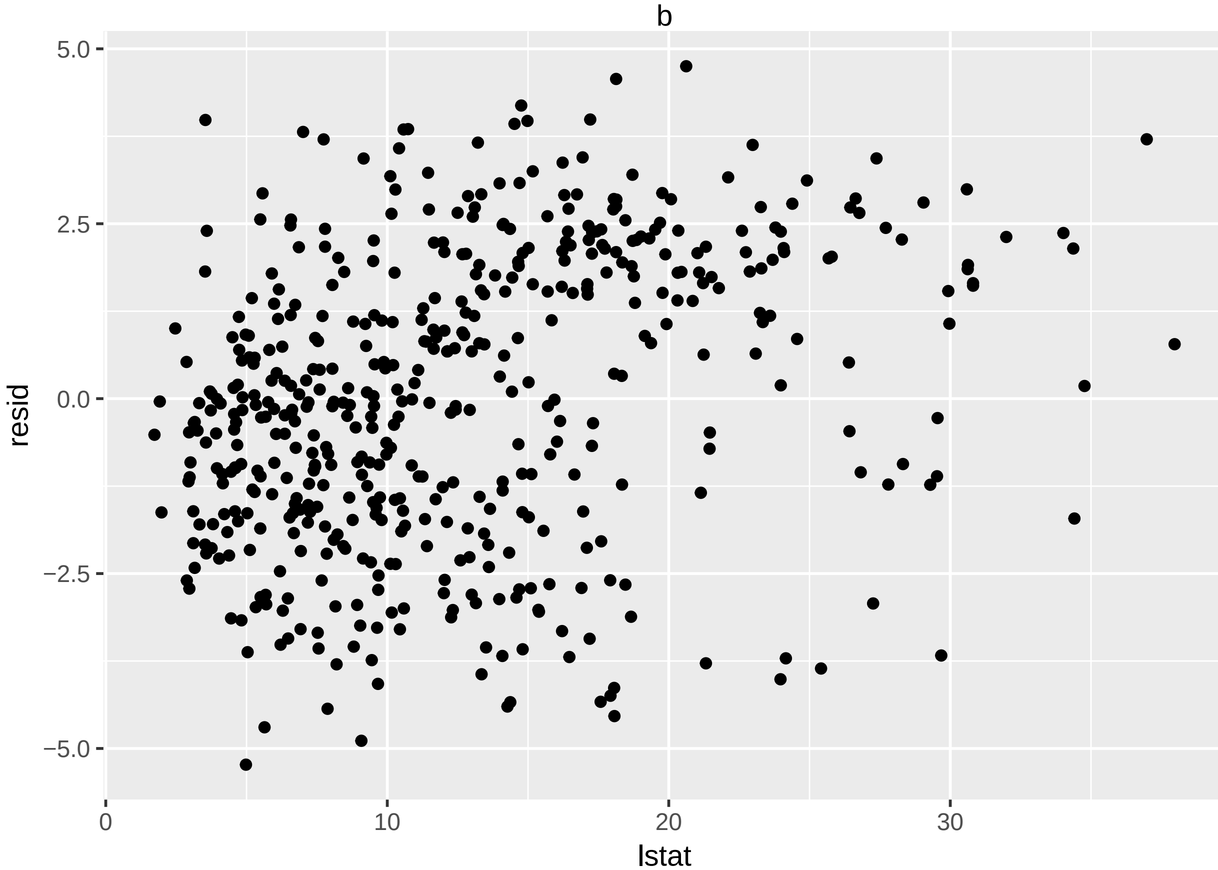
5.0
2.5
0.0
−2.5
−5.0
resid
0
5
10
15
20
25
rad


resid
tax
5.0
2.5
0.0
−2.5
−5.0
200
300
400
500
600
700

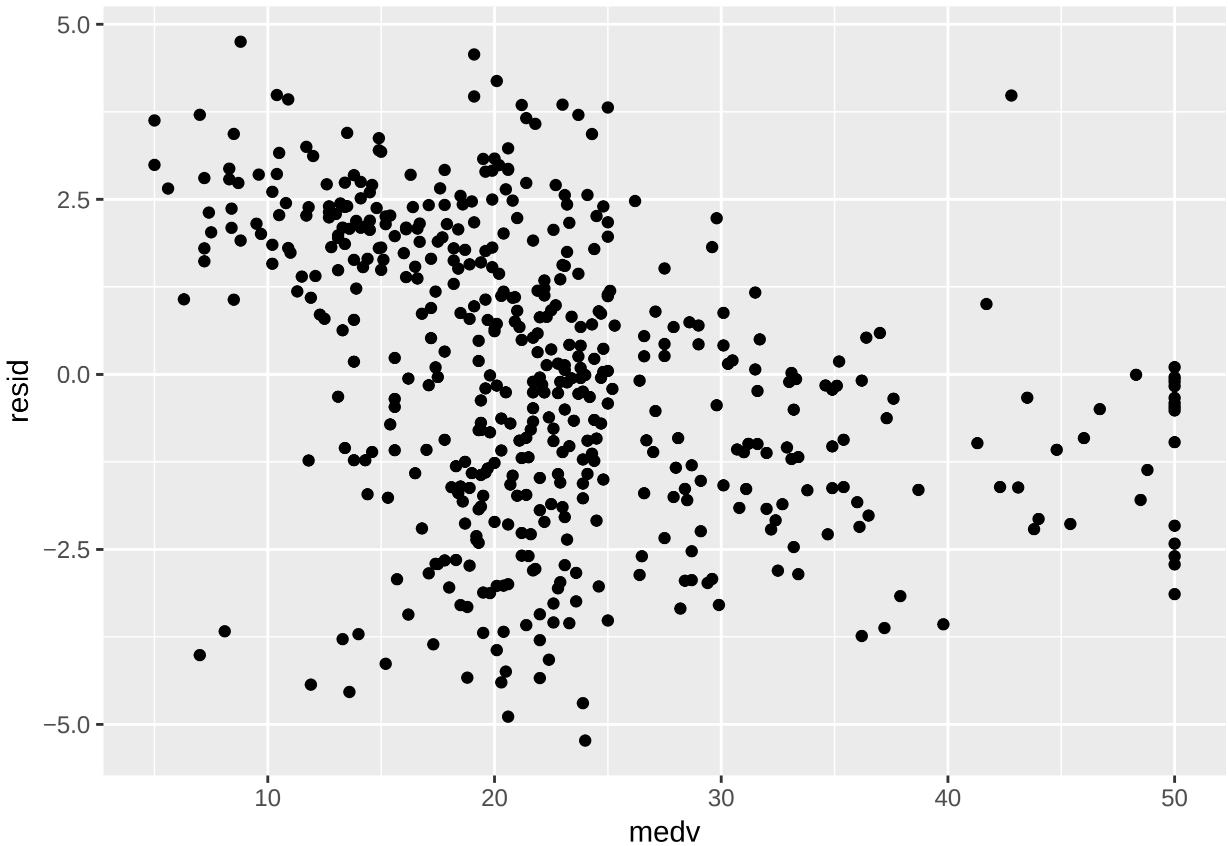

resid
ptratio
5.0
2.5
0.0
−2.5
−5.0
12.5
15.0
17.5
20.0



resid
5.0
2.5
0.0
−2.5
−5.0
0
100
200
300
400
b

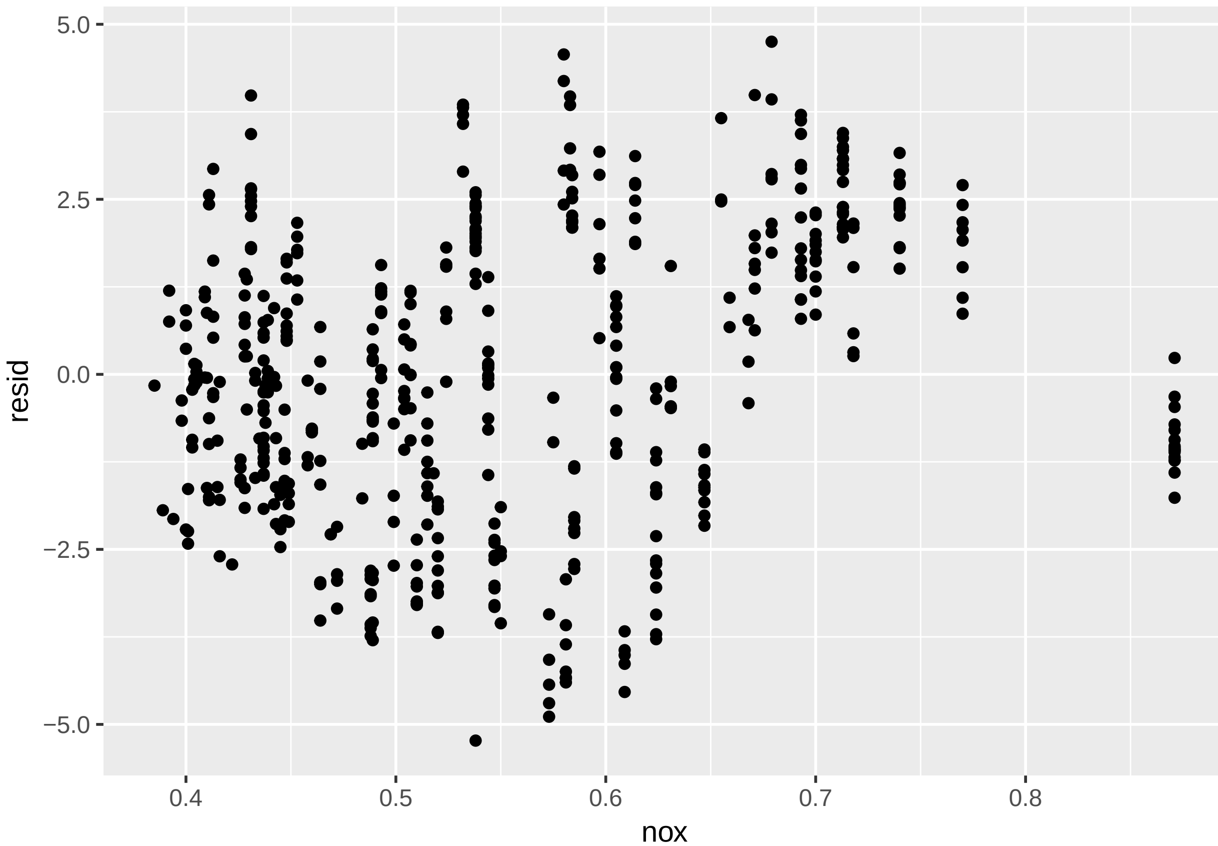


resid
5.0
2.5
0.0
−2.5
−5.0
0
10
20
30
lstat

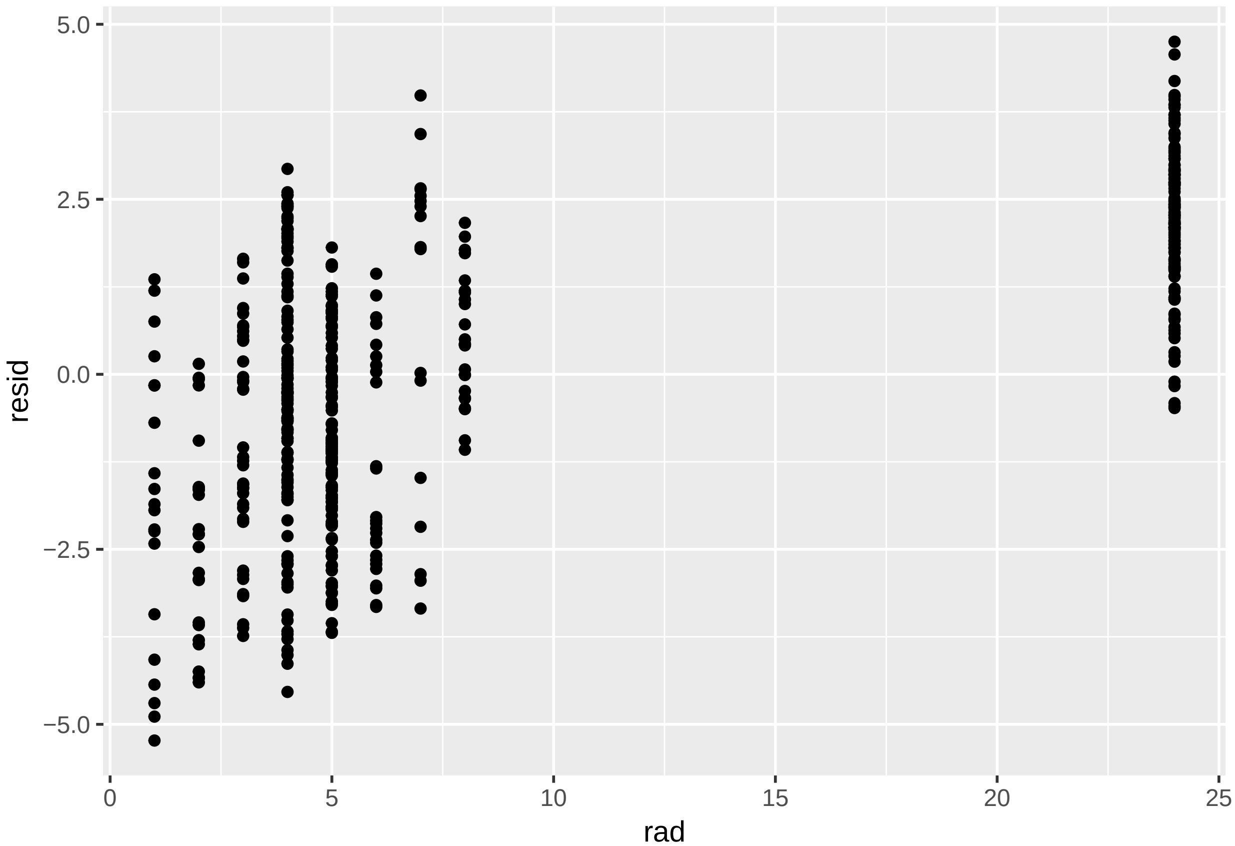


I found out that some plots don't have random distribution over residual.And they are shown below.

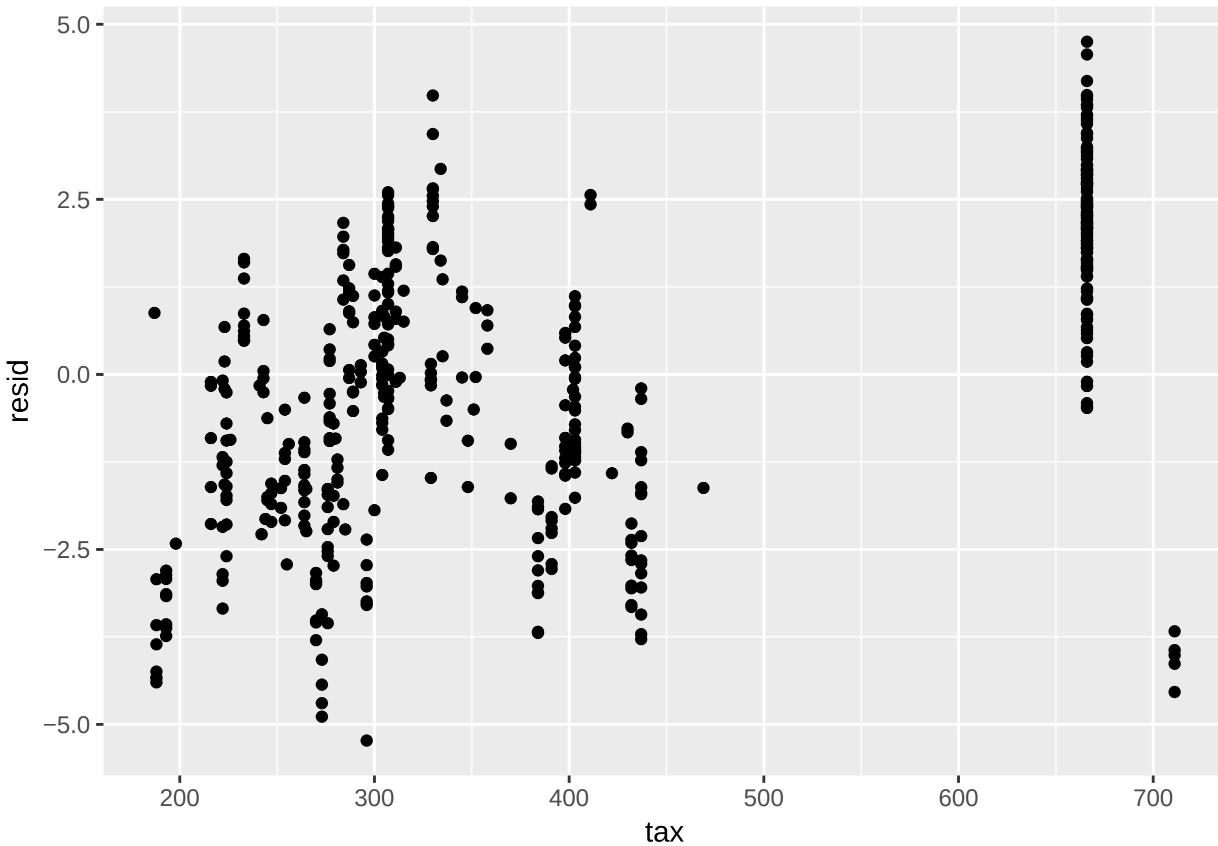
```
data1%>% add_residuals(fit1)%>%
  ggplot()+geom_point(aes(x=nox,y=resid))
```




```
data1%>% add_residuals(fit1)%>%
  ggplot()+geom_point(aes(x=rad,y=resid))
```




```
data1%>% add_residuals(fit1)%>%
  ggplot()+geom_point(aes(x=tax,y=resid))
```



So, I used Adjusted R-square to find the optimal combination.

```
fit2 <- lm(log2(crim)~log2(dis)+rad,data=data1)
summary(fit2)
```

```
##
## Call:
## lm(formula = log2(crim) ~ log2(dis) + rad, data = data1)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.7372 -0.9462 -0.0638  0.8247  3.3859
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.642453   0.211964  -3.031  0.00256 **
## log2(dis)   -1.552176   0.088618 -17.515  < 2e-16 ***
## rad          0.227962   0.007922  28.775  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.284 on 503 degrees of freedom
## Multiple R-squared:  0.8312, Adjusted R-squared:  0.8306
## F-statistic:  1239 on 2 and 503 DF,  p-value: < 2.2e-16
```

```
fit3 <- lm(log2(crim)~log2(dis)+nox,data=data1)
summary(fit3)
```

```
##
## Call:
## lm(formula = log2(crim) ~ log2(dis) + nox, data = data1)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.5685 -1.4365  0.1292  1.2356  4.7852
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -7.3976     1.0059  -7.354 7.84e-13 ***
## log2(dis)    -1.1432     0.1913  -5.975 4.37e-09 ***
## nox          14.8388     1.2853  11.545  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.857 on 503 degrees of freedom
## Multiple R-squared:  0.647,  Adjusted R-squared:  0.6456
## F-statistic: 460.9 on 2 and 503 DF,  p-value: < 2.2e-16
```

```r
fit4 <- lm(log2(crim)~log2(dis)+tax,data=data1)
summary(fit4)
```

```
##
## Call:
## lm(formula = log2(crim) ~ log2(dis) + tax, data = data1)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -6.6765 -0.9351  0.0991  0.9776  3.2280
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -3.055472   0.357563  -8.545   <2e-16 ***
## log2(dis)   -1.502241   0.108469 -13.849   <2e-16 ***
## tax          0.011034   0.000501  22.024   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.49 on 503 degrees of freedom
## Multiple R-squared:  0.7727, Adjusted R-squared:  0.7718
## F-statistic: 854.8 on 2 and 503 DF,  p-value: < 2.2e-16
```

As a result, fit2 has the greatest Adj R-sqr, 0.83. Therefore, the final regression is: log2(crim)=-2.422157-1.640766Xlog2(crim)+2.199782Xrad

PartB

Problem4

The function is shown below:

```r
library(purrr)
Perform_kcross <- function(formula,dataset,k){
```

```r
  dataset_cv <- crossv_kfold(dataset, k)
  dataset_cv <- dataset_cv %>%
    mutate(fit = map(train, ~ lm(formula,data = .)))%>%
    mutate(rmse = map2_dbl(fit,test,rmse))
    return(mean(dataset_cv$rmse))
}
```

Problem5

```r
set.seed(12)
(Problem1_model=Perform_kcross(log(crim)~log(dis),data1,5))
```

```
## [1] 1.451747
```

```r
(Problem3_model=Perform_kcross(log(crim)~log(dis)+rad,data1,5))
```

```
## [1] 0.8925567
```

The model from problem 3 has a much smaller root-mean-square-error, which is 0.89,while the root-mean-square-error of model from problem 1 is around 1.45. Therefore, model from problem 3 is better due to a smaller error.

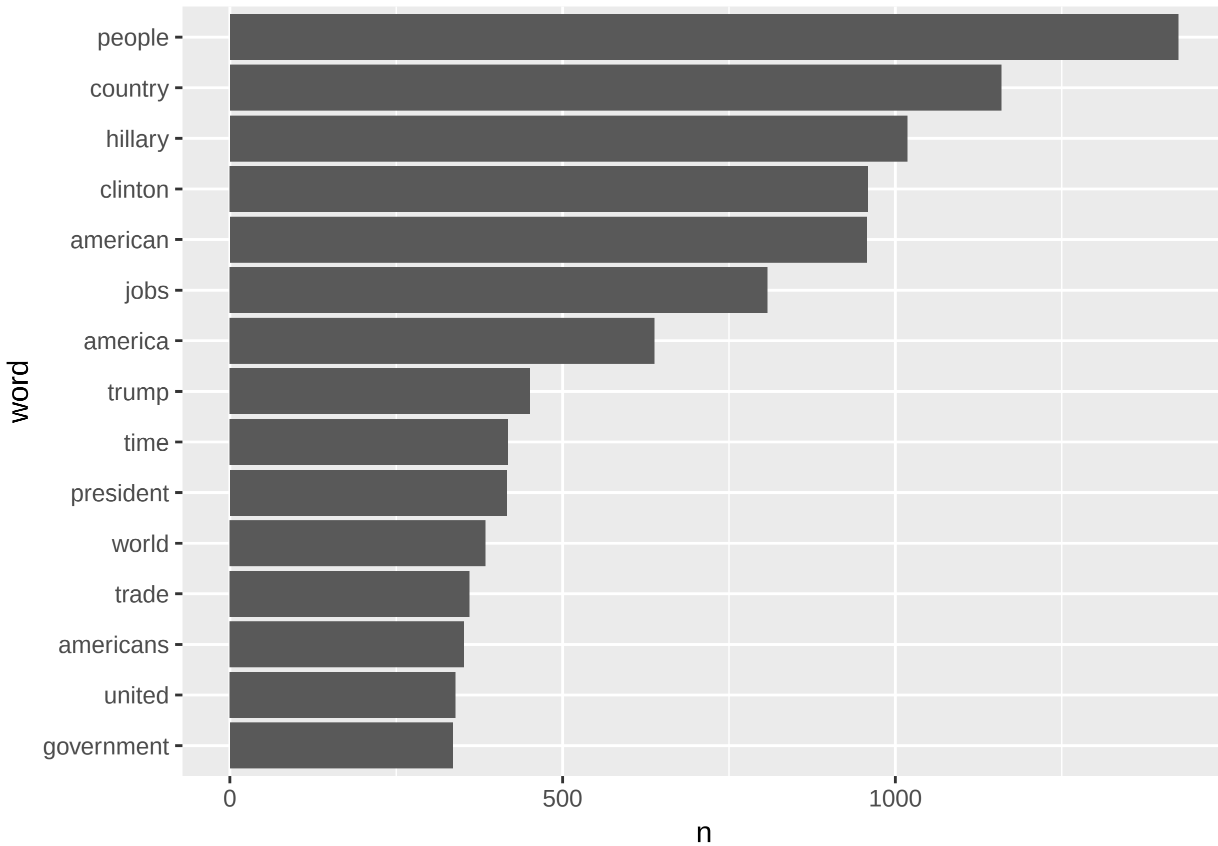
PartC

Problem6

```r
library(tokenizers)
library(tidytext)
text <- readLines("full_speech.txt")
speech <- tibble(line=1:length(text), text=text)
speech_tidy <- speech %>% unnest_tokens(word, text)

speech_tidy %>%
  filter(word!='applause')%>%
  anti_join(stop_words, by="word") %>% count(word, sort=TRUE) %>%
  top_n(15) %>%
  mutate(word = reorder(word, n)) %>% ggplot(aes(x=word, y=n)) +
  geom_col() +
  coord_flip()
```

```
## Selecting by n
```

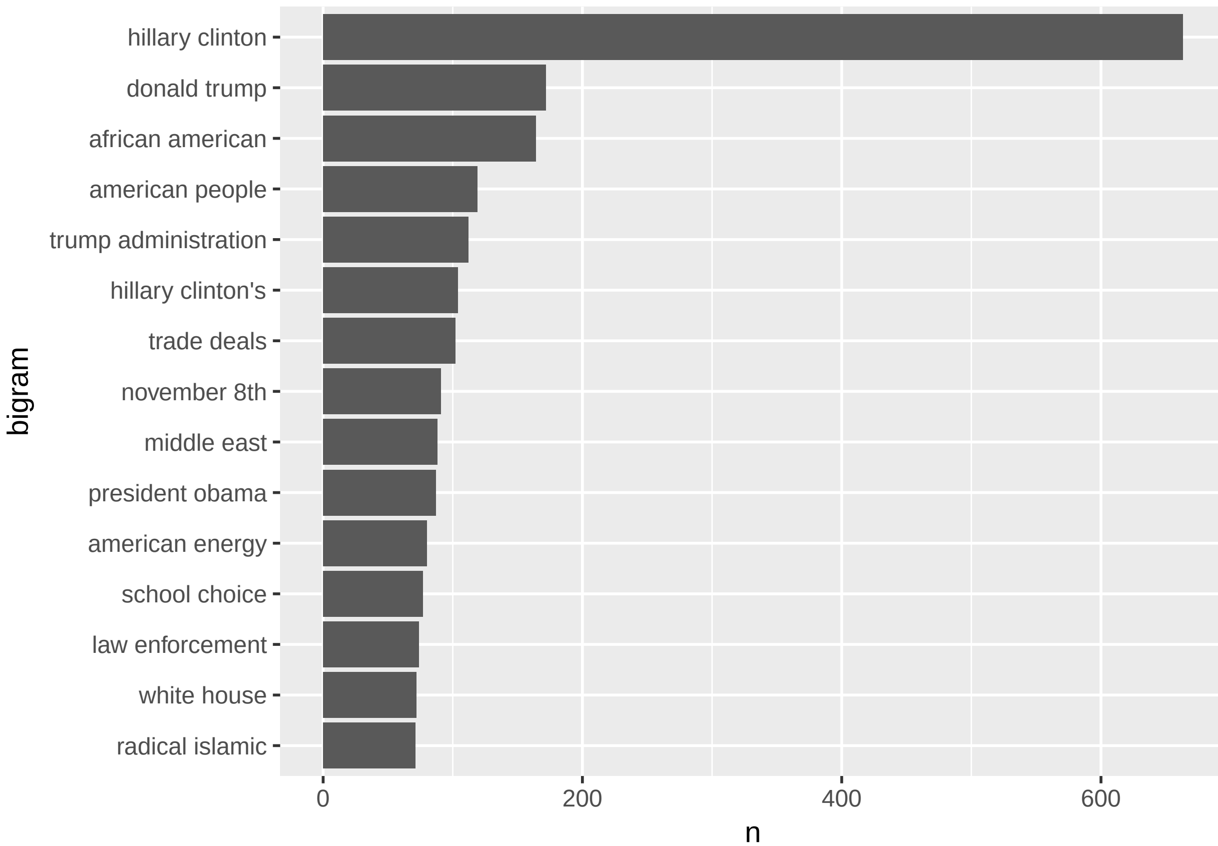


Problem7

```
speech_bi <- speech%>%
  unnest_tokens(bigram,text, token = "ngrams", n = 2)%>%
  separate(bigram, c("word1", "word2"), sep = " ")%>%
  filter(!word1 %in% stop_words$word) %>%
  filter(!word2 %in% stop_words$word) %>%
  filter(word1!='applause') %>%
  filter(word2!='applause')

speech_bi_1 <- speech_bi%>%unite(bigram, word1, word2, sep = " ")%>%
  count(bigram,sort=TRUE)

speech_bi_1%>%
  filter(n>=speech_bi_1$n[15])%>%
  mutate(bigram = reorder(bigram, n))%>%
  ggplot(aes(x=bigram,y=n))+geom_col()+
  coord_flip()
```



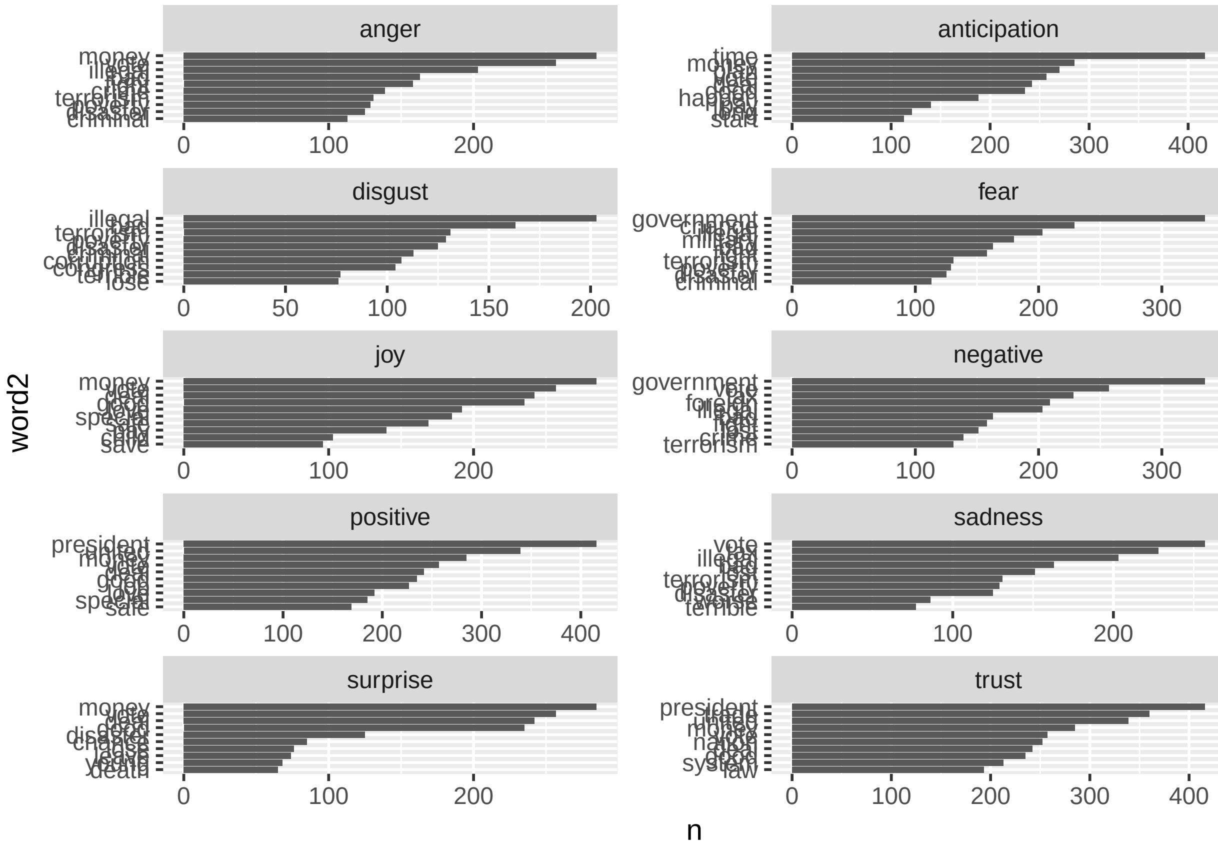
Problem8

```
speech_bi_sentiment <- speech%>%
  unnest_tokens(bigram,text, token = "ngrams", n = 2)%>%
  separate(bigram, c("word1", "word2"), sep = " ") %>%
  filter(word1!='not')%>%
  filter(word1!='no')%>%
  filter(word1!='never')

speech_bi_sentiment_word2 <- speech_bi_sentiment%>%
  transmute(word2=word2)%>%
  filter(word2!='applause')%>%
  filter(word2!='trump')%>%
  inner_join(get_sentiments("nrc"),by=c('word2'='word'))%>%
  count(word2,sentiment,sort=TRUE)

speech_bi_sentiment_word2%>%
  group_by(sentiment)%>%
  top_n(10)%>%
  ungroup()%>%
  mutate(word2=reorder(word2,n))%>%
  ggplot(aes(x=word2,y=n))+
  geom_col(show.legend=FALSE)+
  coord_flip()+
  facet_wrap(~sentiment,ncol=2,scales = "free")
```

```
## Selecting by n
```



Problem9

```
tidy_corpus <- function(src){
  src_1=tibble(token=as.character(src),
               line=1:length(src))
  src_2 <- src_1 %>% unnest_tokens(word, token)
  class(src_2)=c('tidy_corpus','tbl_df','tbl','data.frame')
  return(src_2)
}

tidy_corpus_plot <- function(src){
  b <- tidy_corpus(src)
  b %>%
    anti_join(stop_words, by="word") %>% count(word, sort=TRUE) %>%
    top_n(10) %>%
    mutate(word = reorder(word, n)) %>% ggplot(aes(x=word, y=n)) +
    geom_col() +
    coord_flip()
}

tidy_corpus(speech)
```

```
## # A tibble: 236,292 x 2
##     line  word
```

```
##    <int> <chr>
##  1     1     1
##  2     1    74
##  3     2     c
##  4     2 trump
##  5     2   wow
##  6     2  whoa
##  7     2  that
##  8     2    is
##  9     2  some
## 10     2 group
## # ... with 236,282 more rows
```

```r
tidy_corpus_plot(speech)
```

```
## Selecting by n
```