STAT 435 HW 4

Pat McCornack

2022-11-30

Question 1 - (ISLR Ch. 7 - Q7)

Initial EDA

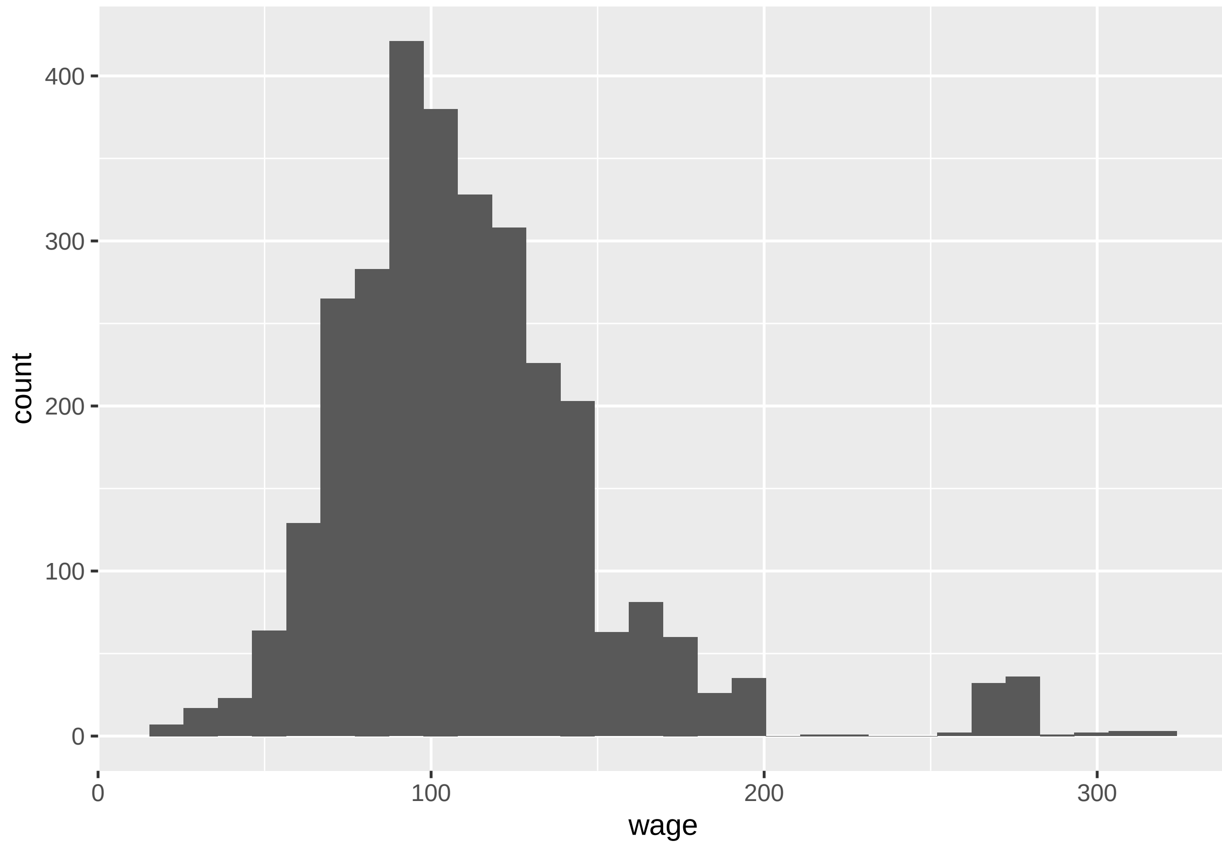
First I did some initial exploration variance of variables as well as covariance among variables using visualization and summarization. One thing to note is that the demographics of the data are highly skewed towards men that are married and white.

A couple of trends of note among variables is that age and marital status are strongly correlated as are job class and education. This is of note because there appears to be a trend between job class and wage where information jobs tend to pay higher on average than industrial jobs.

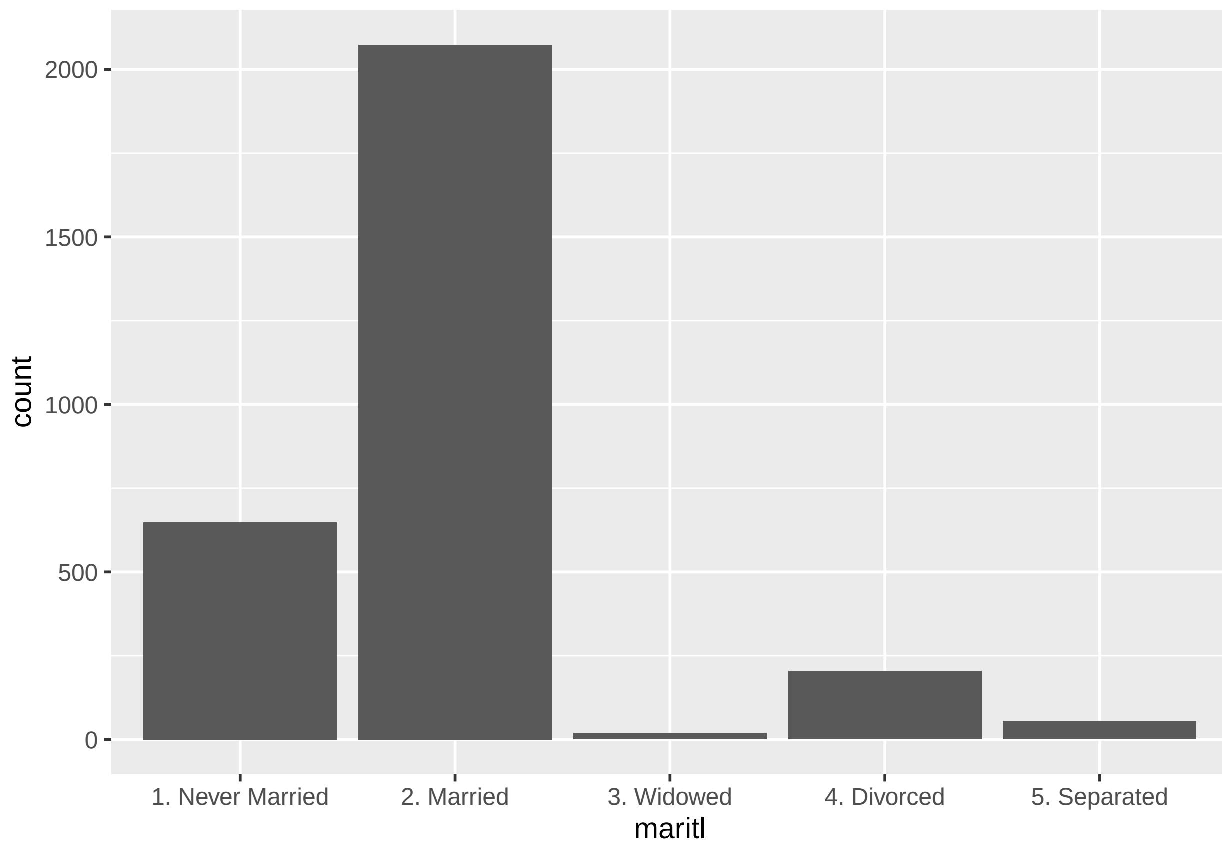
```
library(ISLR)
library(ggplot2)
attach(Wage)
data(Wage)
summary(Wage)
```

```
##       year           age                     maritl           race
##  Min.   :2003   Min.   :18.00   1. Never Married: 648   1. White:2480
##  1st Qu.:2004   1st Qu.:33.75   2. Married      :2074   2. Black: 293
##  Median :2006   Median :42.00   3. Widowed      :  19   3. Asian: 190
##  Mean   :2006   Mean   :42.41   4. Divorced     : 204   4. Other:  37
##  3rd Qu.:2008   3rd Qu.:51.00   5. Separated    :  55
##  Max.   :2009   Max.   :80.00
##
##               education                     region               jobclass
##  1. < HS Grad      :268   2. Middle Atlantic   :3000   1. Industrial :1544
##  2. HS Grad        :971   1. New England       :   0   2. Information:1456
##  3. Some College   :650   3. East North Central:   0
##  4. College Grad   :685   4. West North Central:   0
##  5. Advanced Degree:426   5. South Atlantic    :   0
##                           6. East South Central:   0
##                           (Other)              :   0
##             health      health_ins      logwage          wage
##  1. <=Good     : 858   1. Yes:2083   Min.   :3.000   Min.   : 20.09
##  2. >=Very Good:2142   2. No : 917   1st Qu.:4.447   1st Qu.: 85.38
##                                      Median :4.653   Median :104.92
##                                      Mean   :4.654   Mean   :111.70
##                                      3rd Qu.:4.857   3rd Qu.:128.68
##                                      Max.   :5.763   Max.   :318.34
##
```

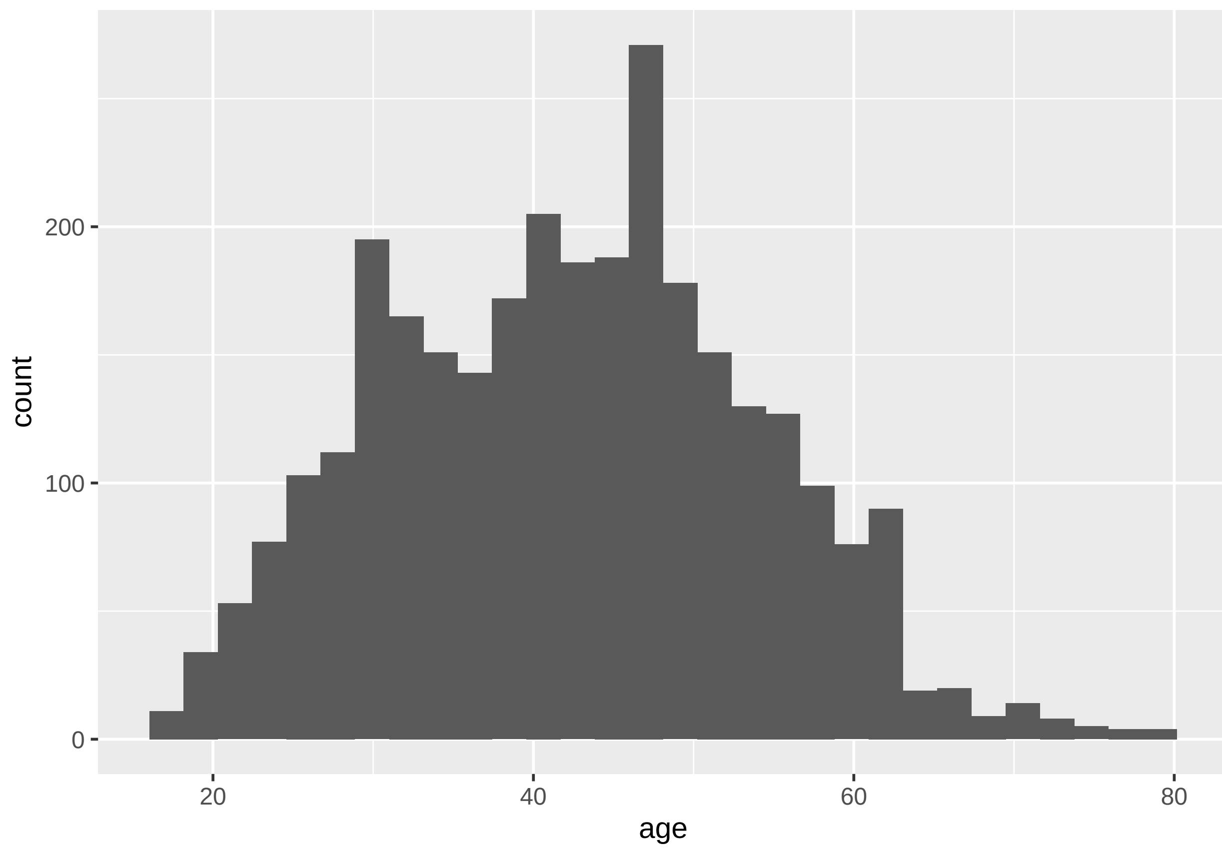
```
set.seed(1234)

# Exploration of variance within variables

ggplot(data = Wage) +
  geom_histogram(aes(x=wage))
```



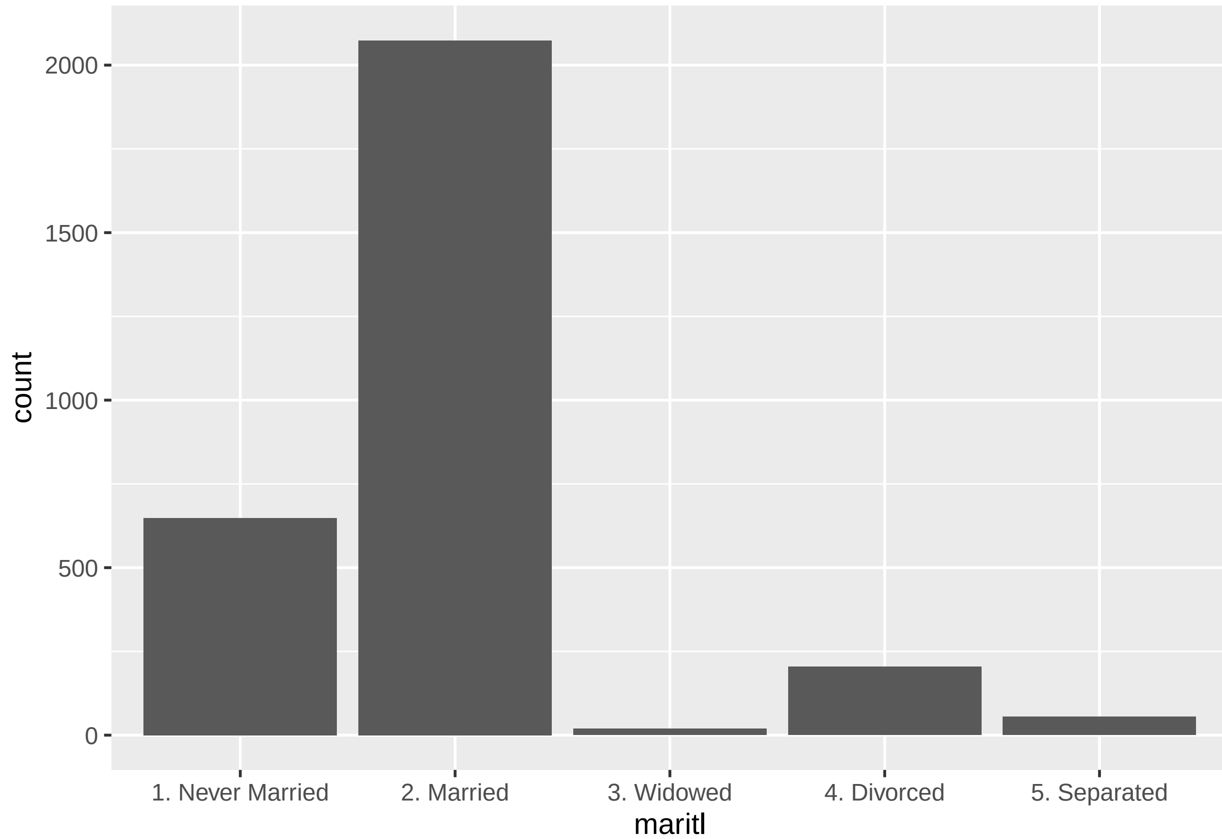

```
ggplot(data = Wage, aes(x = maritl)) +
  geom_bar()
```



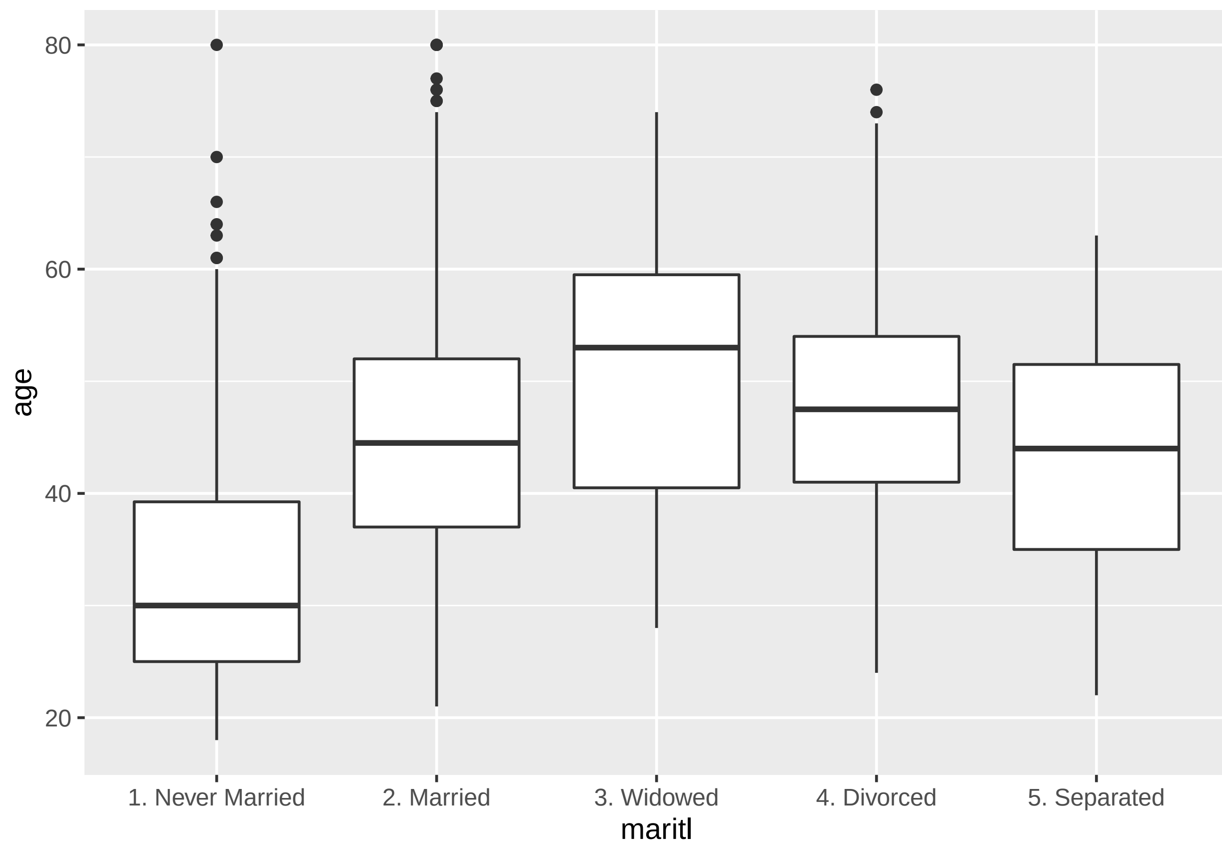

```
ggplot(data = Wage, aes(x = age)) +
  geom_histogram()
```



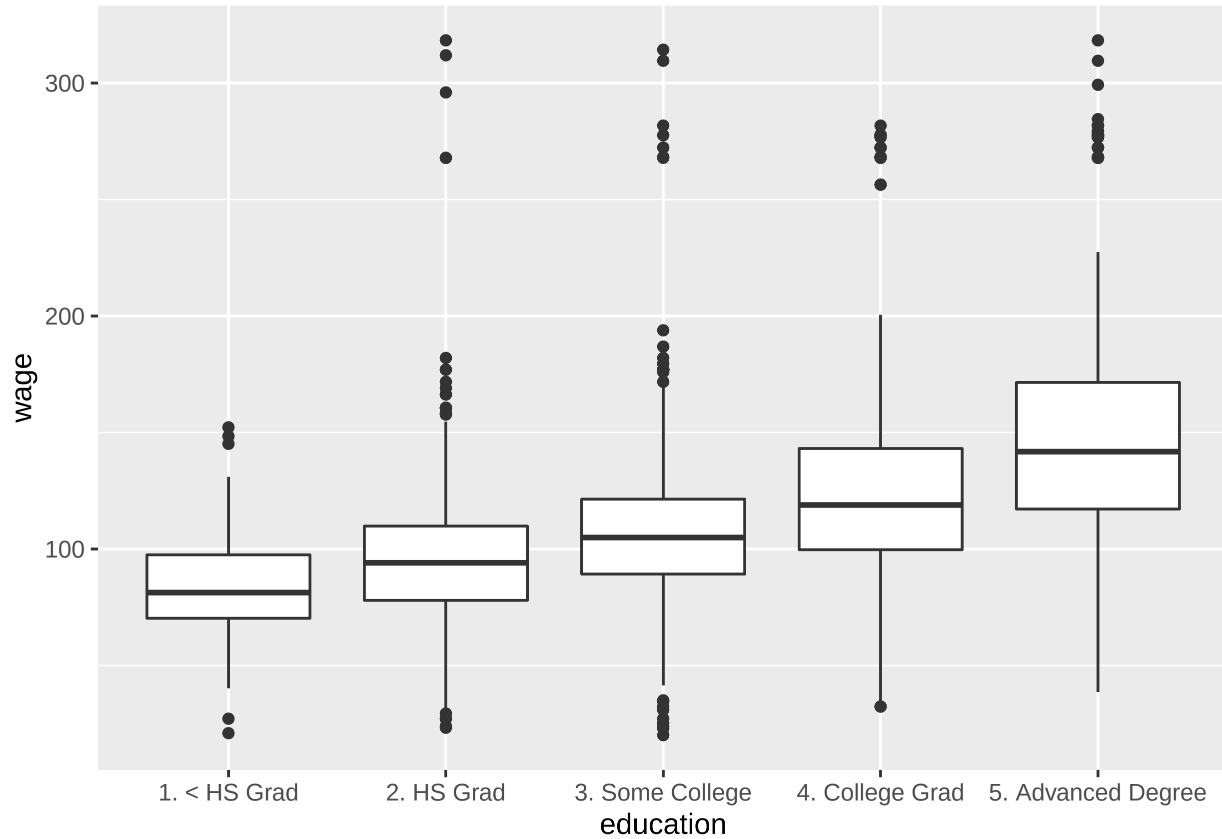

```
ggplot(data = Wage, aes(x = maritl)) +
  geom_bar()
```



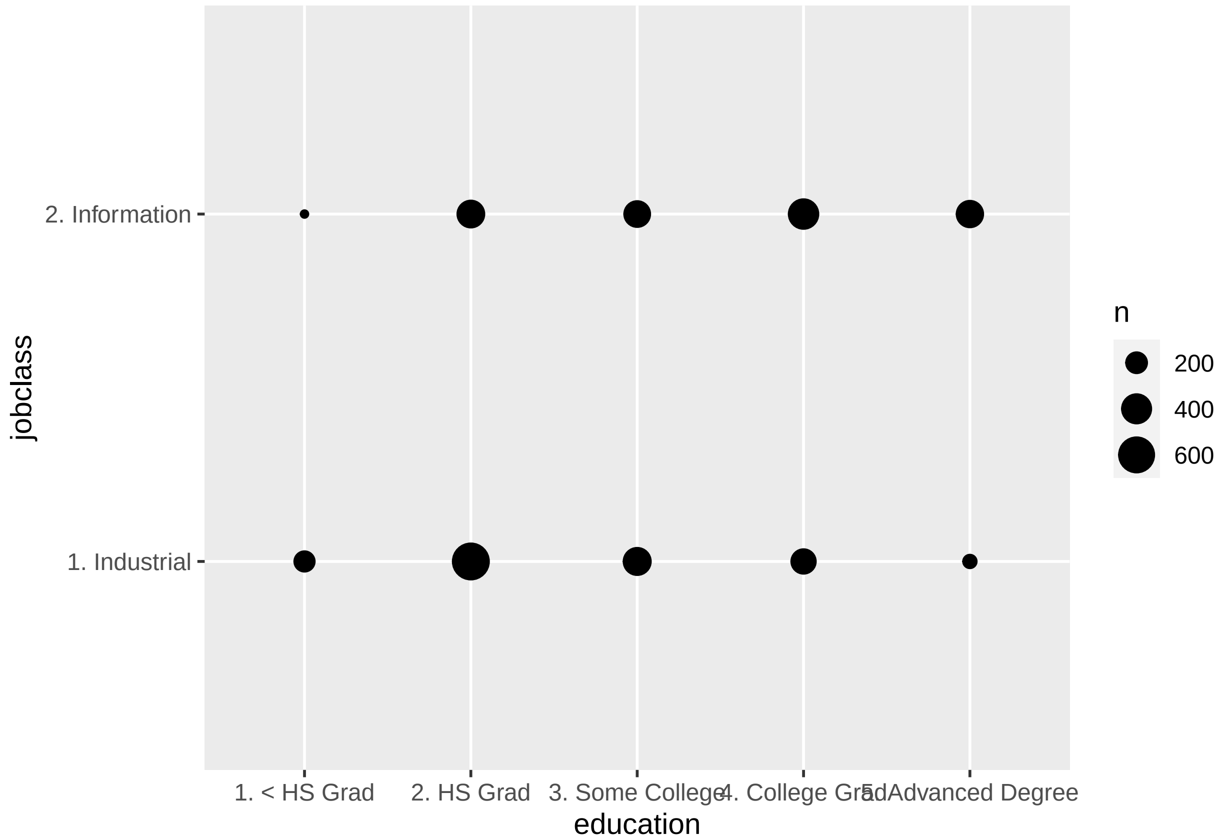

```
# Exploration of covariance
ggplot(data = Wage, aes(x = maritl, y = age)) +
  geom_boxplot()
```



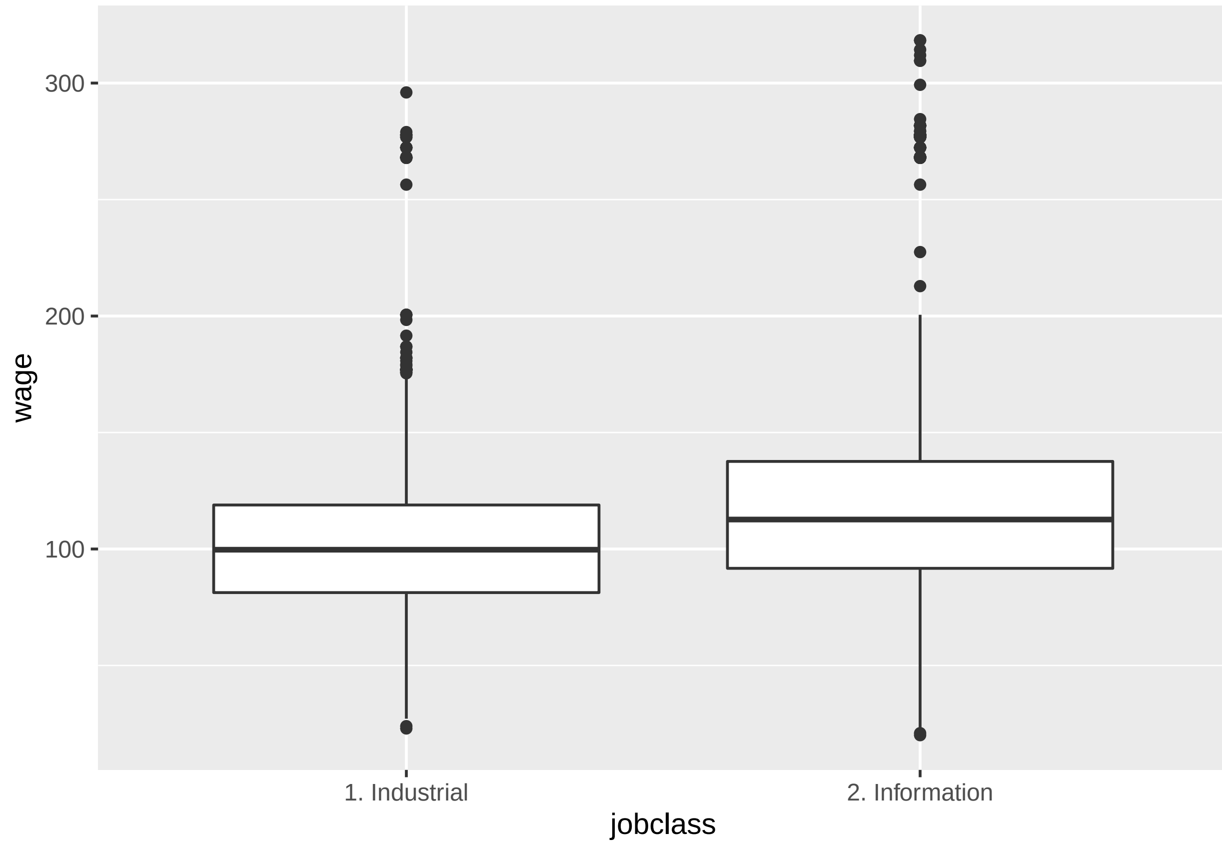

```
ggplot(data = Wage) +
  geom_boxplot(aes(x=education, y=wage))
```



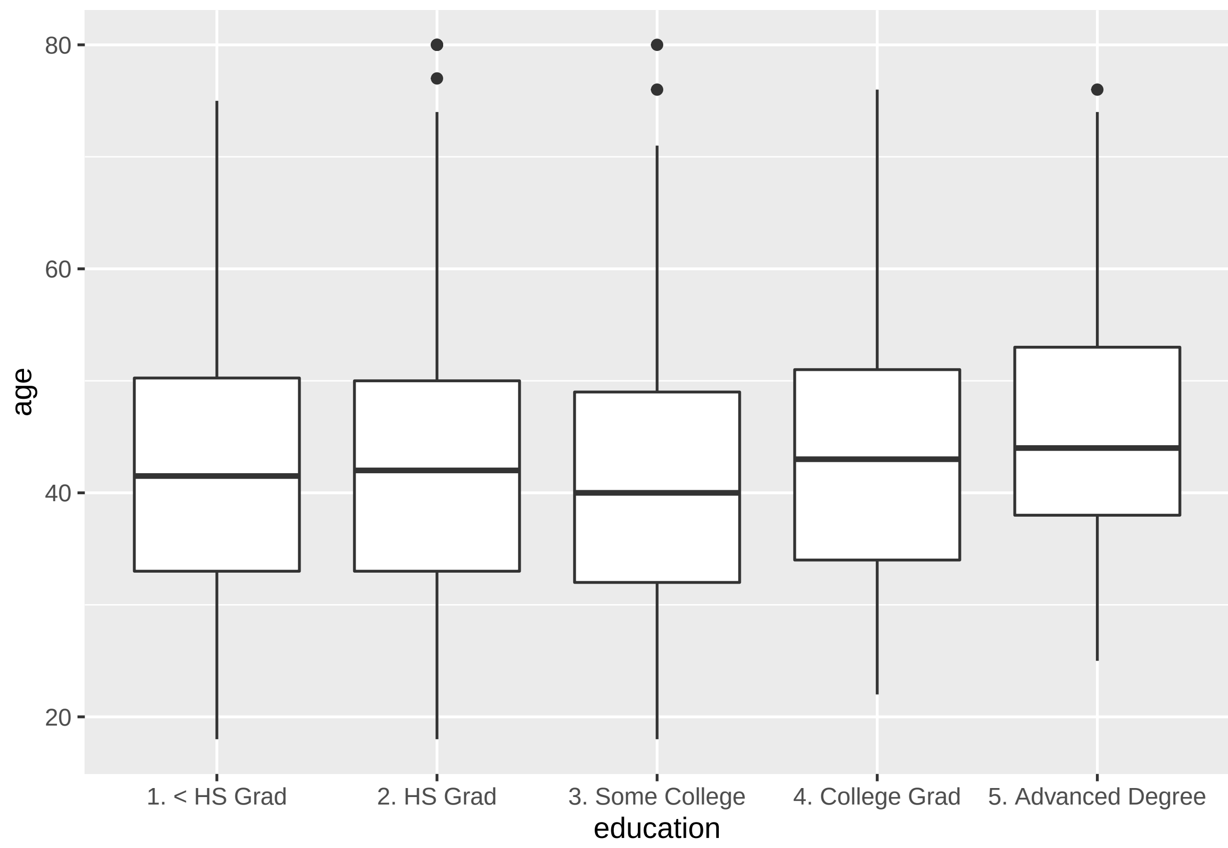

```
ggplot(data = Wage) +
  geom_count(aes(x=education, y=jobclass))
```



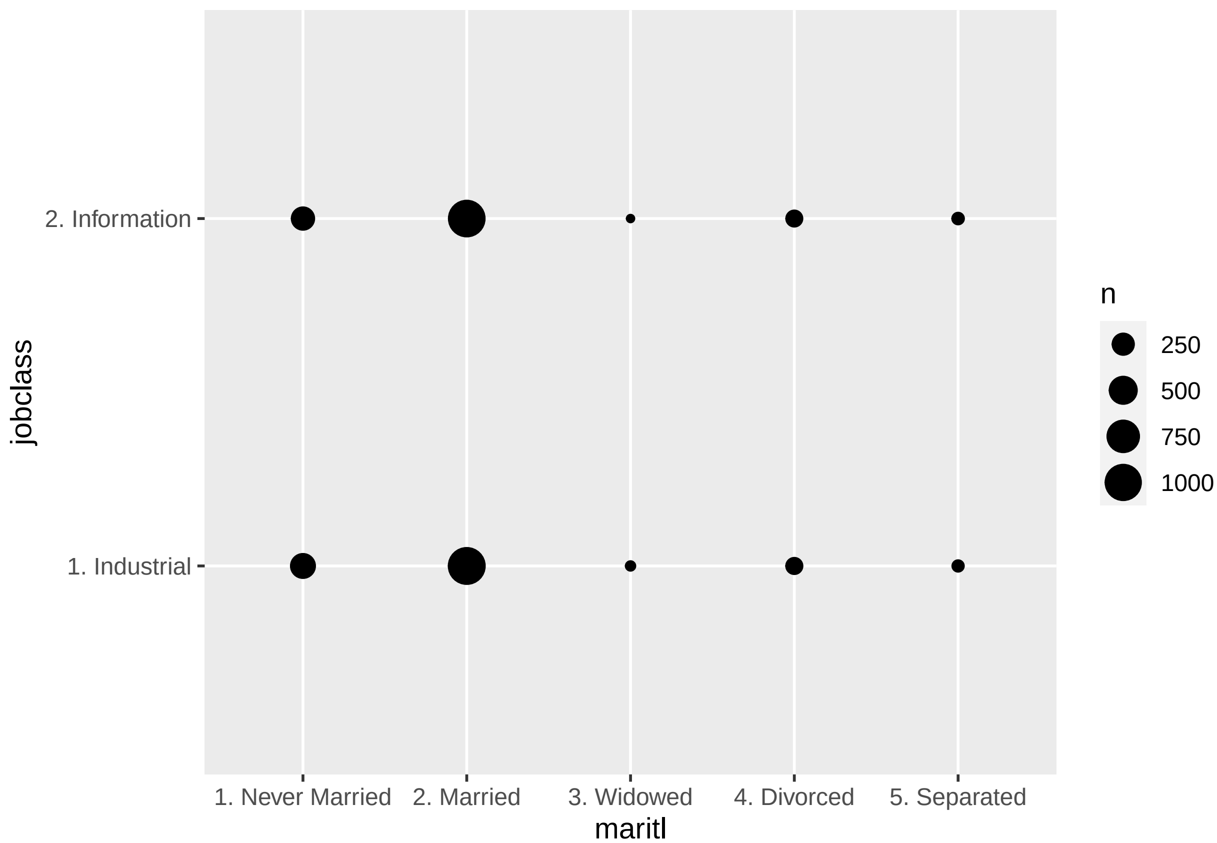

```
ggplot(data = Wage, aes(x=jobclass, y=wage)) +
  geom_boxplot()
```




```r
ggplot(data = Wage, aes(x = education, y = age)) +
  geom_boxplot()
```




```
ggplot(data = Wage) +
  geom_count(aes(x = maritl, y = jobclass))
```

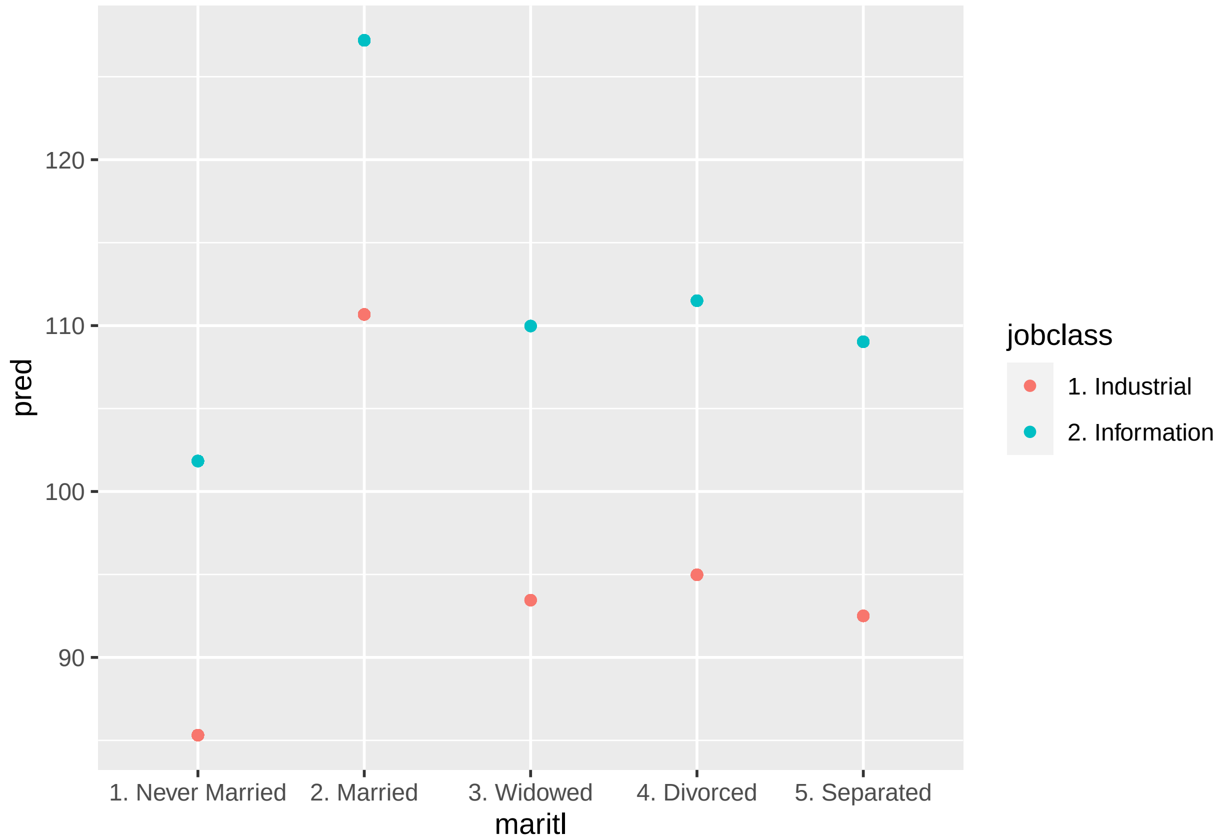


Modeling

There doesn't appear to be any colinearity between marital status and job class, so we will use these variables as predictors for wage. We use a generalized additive model to fit these qualitative predictors for the wage data. The result is a model that predicts wage based on marital status and job class. The model was then used to make predictions using the wage dataset. These predictions confirm our early hypothesis that information workers tend to make more than industrial workers and also show that married individuals make more than the other marital statuses across both categories. However, it's important to note that this may be because married individuals are typically in the age bracket where they have the highest earning potential. The boxplot shows that individuals that have never married tend to be much younger, and we know from earlier analyses that younger people tend to earn less. The same is true of the population of retirement age which the average of the widowed group tends toward. Additionally, much less data is available for the widowed/divorced/separated categories.

I also created another generalized additive model to predict wage using age and job class with similar conclusions to be drawn as the above and prior analyses.

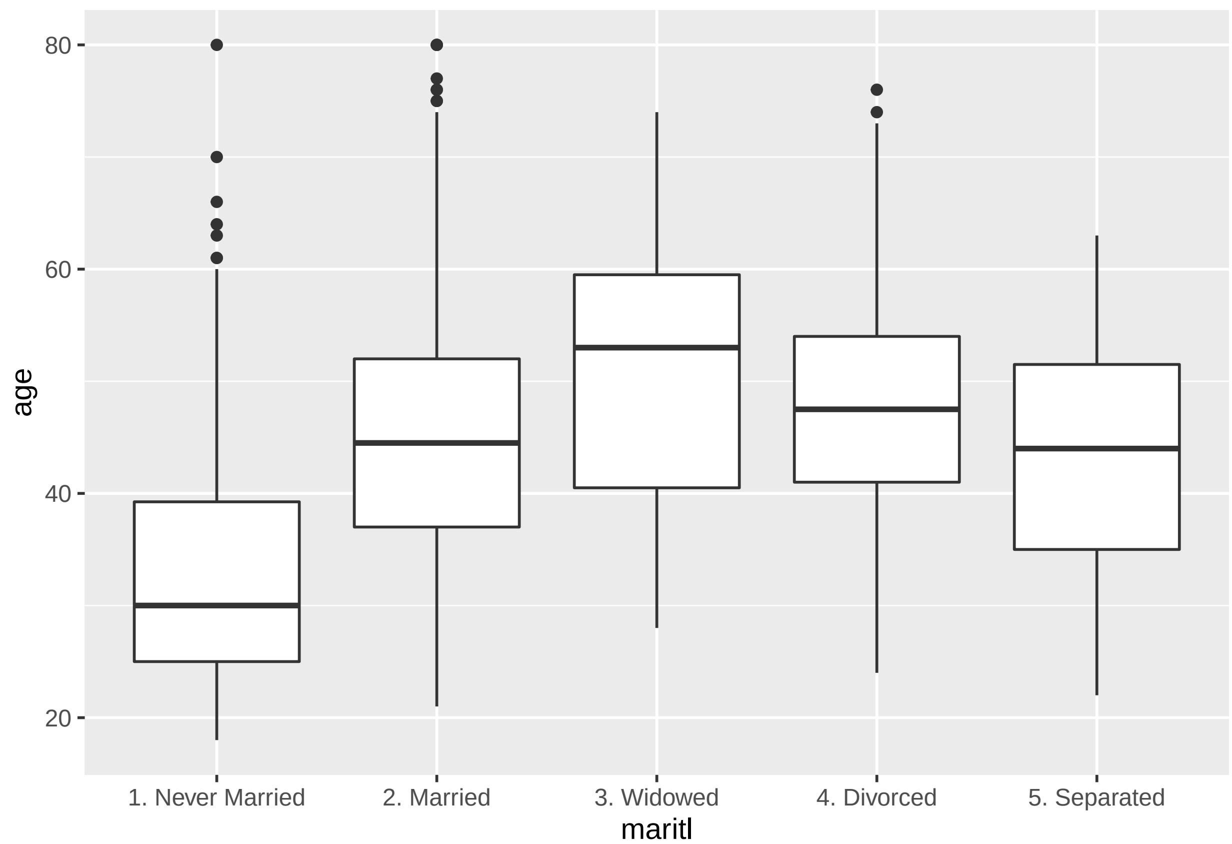
```
library(gam)

fit <- gam(wage ~ maritl + jobclass, data = Wage)
pred <- predict(fit, newdata = Wage$jobclass)

ggplot(data = Wage) +
  geom_point(aes(x = maritl, y = pred, color=jobclass))
```



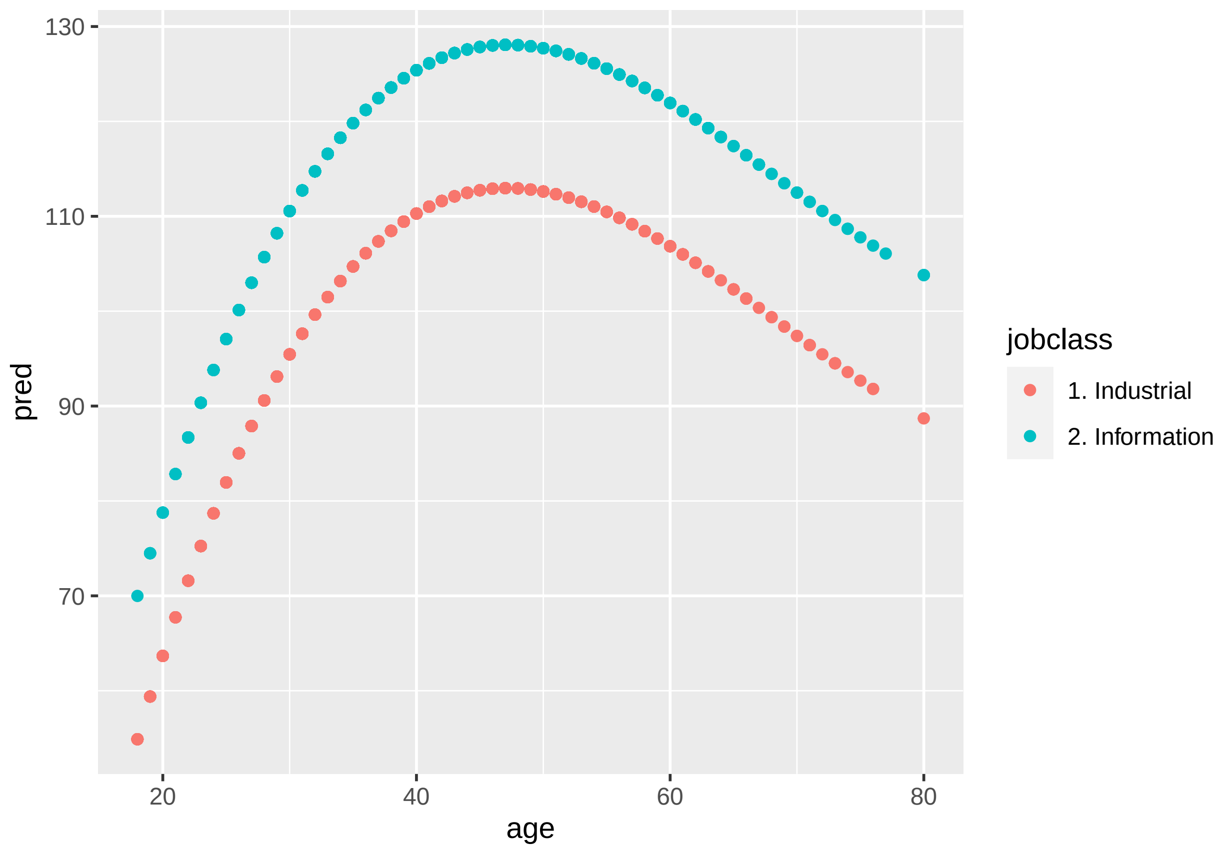

```r
MSE <- sum((Wage$wage - pred)^2) / nrow(Wage)
MSE
```

```
## [1] 1551.584
```

```r
ggplot(data = Wage, aes(x = maritl, y = age)) +
  geom_boxplot()
```




```r
# Including age and jobclass
fit <- gam(wage ~ poly(age, 3) + jobclass, data = Wage)
pred <- predict(fit, newdata = Wage$jobclass)

ggplot(data = Wage) +
  geom_point(aes(x = age, y = pred, color=jobclass))
```




```
MSE <- sum((Wage$wage - pred)^2) / nrow(Wage)
MSE
```

```
## [1] 1536.234
```

Question 2 - (ISLR Ch 7. Q8)

After using ggpairs() to get an overview of the data, we recognize that there are few relationships that appear to be non-linear. In particular we first predict mpg using the horsepower and number of cylinders as well as the weight and number of cylinders. We fit polynomial models of the third degree to both of these relationships. Both relationships show that there is indeed a non-linear fit to the data and that the mpg value drops off as both weight and the number of cylinders increases.

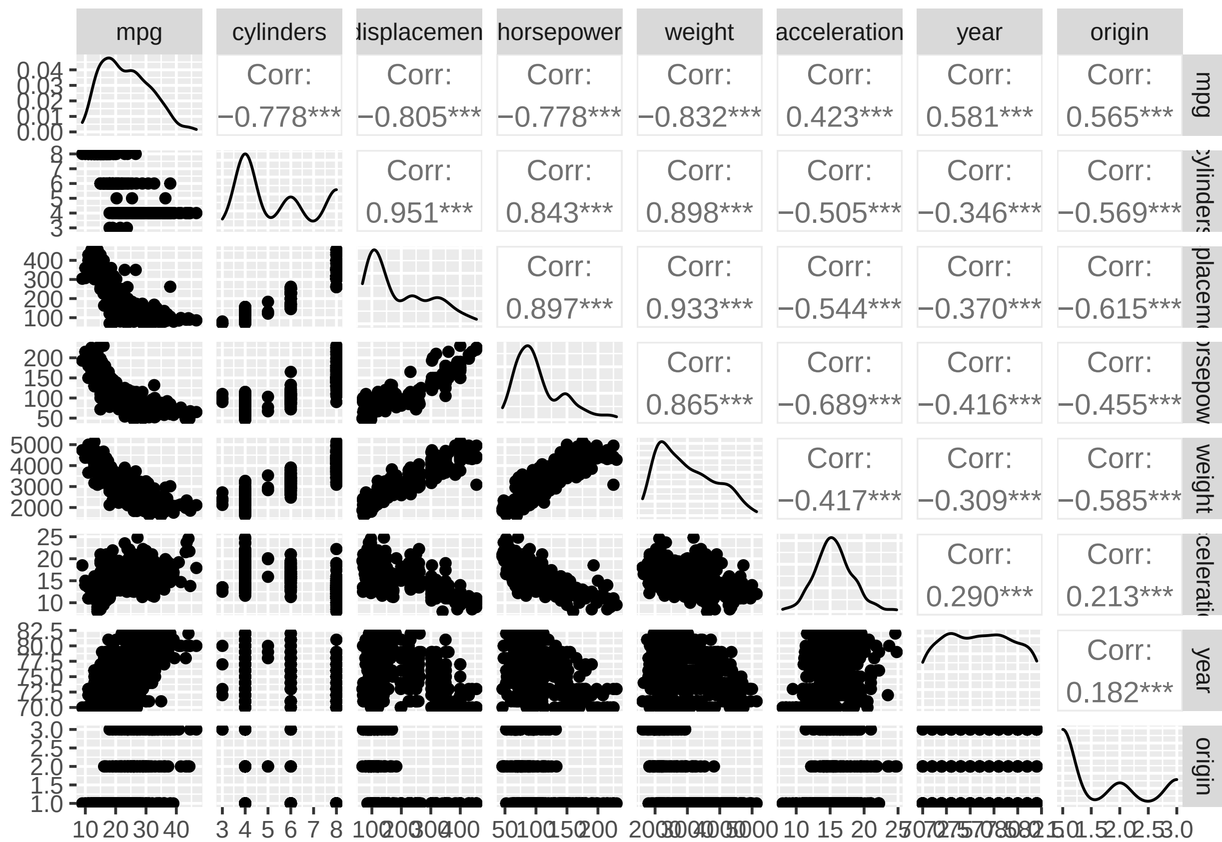
We also create a model to predict acceleration from a combination of weight and number of cylinders. This is a much more interesting plot because the plot shows the interaction between the predictor terms. We would expect a lighter vehicle with more cylinders to have a higher acceleration but for 4 and 6 cylinders we see that the vehicles with the highest acceleration in each bracket are actually the heaviest ones while for 8 cylinders the curve is convex with the highest acceleration in the middle of the bracket. These effects would not be captured by a linear model.

```
attach(Auto)
library(GGally)
data(Auto)
summary(Auto)
```

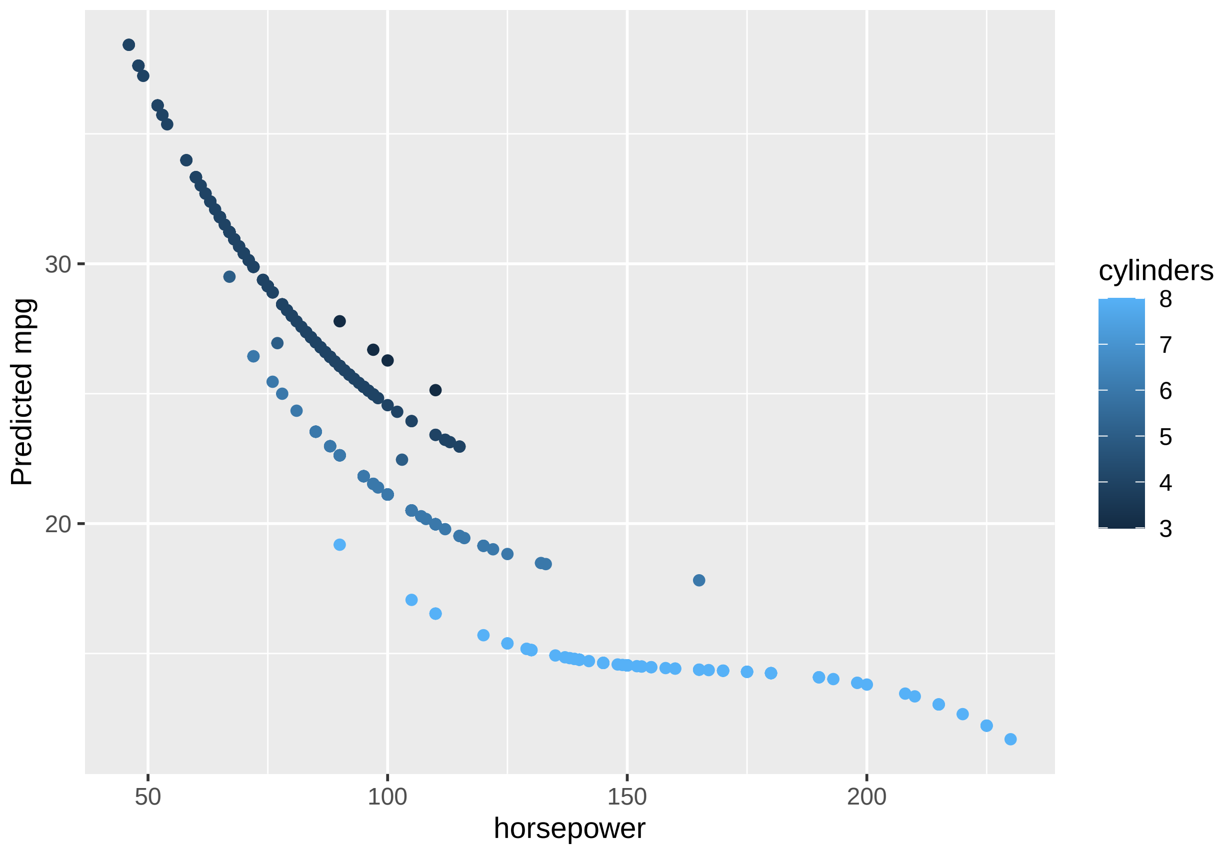
```
##       mpg          cylinders      displacement     horsepower        weight
##  Min.   : 9.00   Min.   :3.000   Min.   : 68.0   Min.   : 46.0   Min.   :1613
```

```
##  1st Qu.:17.00   1st Qu.:4.000   1st Qu.:105.0   1st Qu.: 75.0   1st Qu.:2225
##  Median :22.75   Median :4.000   Median :151.0   Median : 93.5   Median :2804
##  Mean   :23.45   Mean   :5.472   Mean   :194.4   Mean   :104.5   Mean   :2978
##  3rd Qu.:29.00   3rd Qu.:8.000   3rd Qu.:275.8   3rd Qu.:126.0   3rd Qu.:3615
##  Max.   :46.60   Max.   :8.000   Max.   :455.0   Max.   :230.0   Max.   :5140
##
##   acceleration        year           origin                      name
##  Min.   : 8.00   Min.   :70.00   Min.   :1.000   amc matador       :  5
##  1st Qu.:13.78   1st Qu.:73.00   1st Qu.:1.000   ford pinto        :  5
##  Median :15.50   Median :76.00   Median :1.000   toyota corolla    :  5
##  Mean   :15.54   Mean   :75.98   Mean   :1.577   amc gremlin       :  4
##  3rd Qu.:17.02   3rd Qu.:79.00   3rd Qu.:2.000   amc hornet        :  4
##  Max.   :24.80   Max.   :82.00   Max.   :3.000   chevrolet chevette:  4
##                                                  (Other)           :365
```

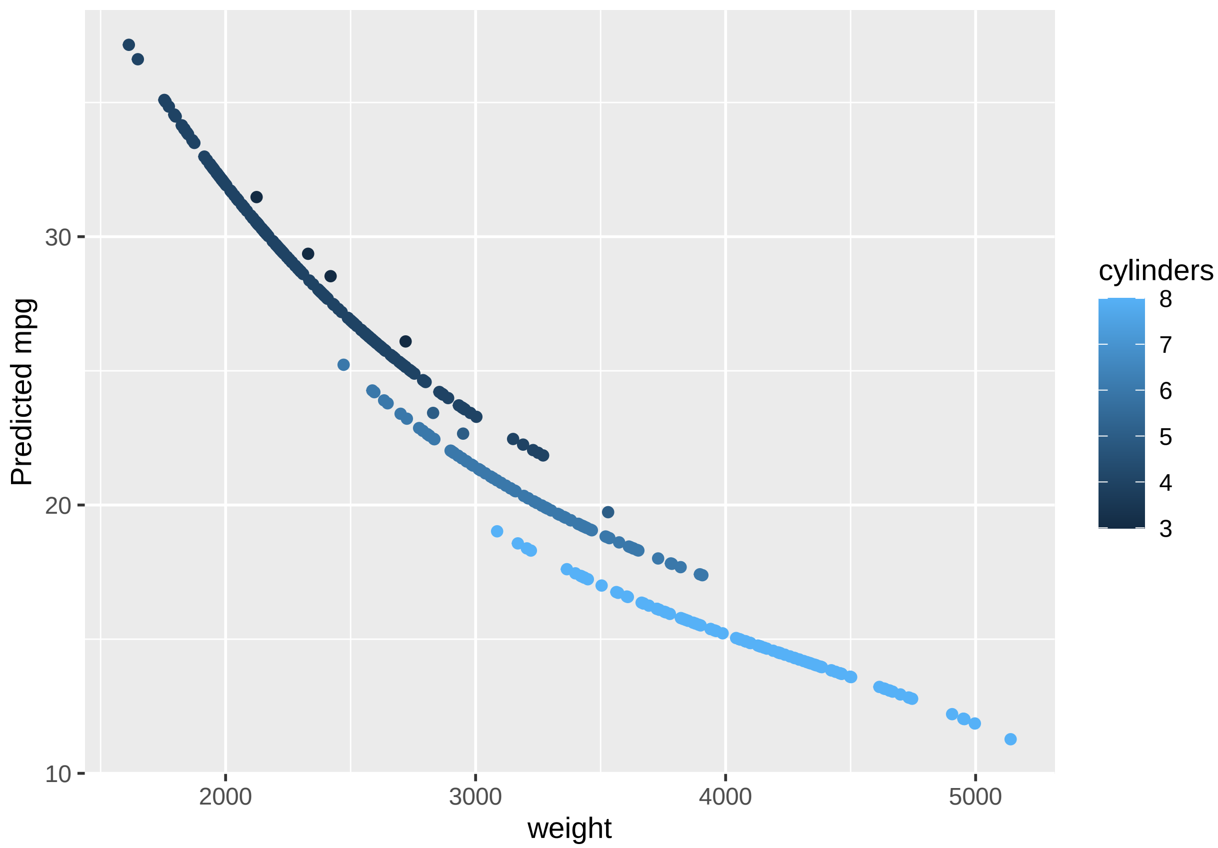
```r
(ggp <- ggpairs(Auto[ ,-9]))
```



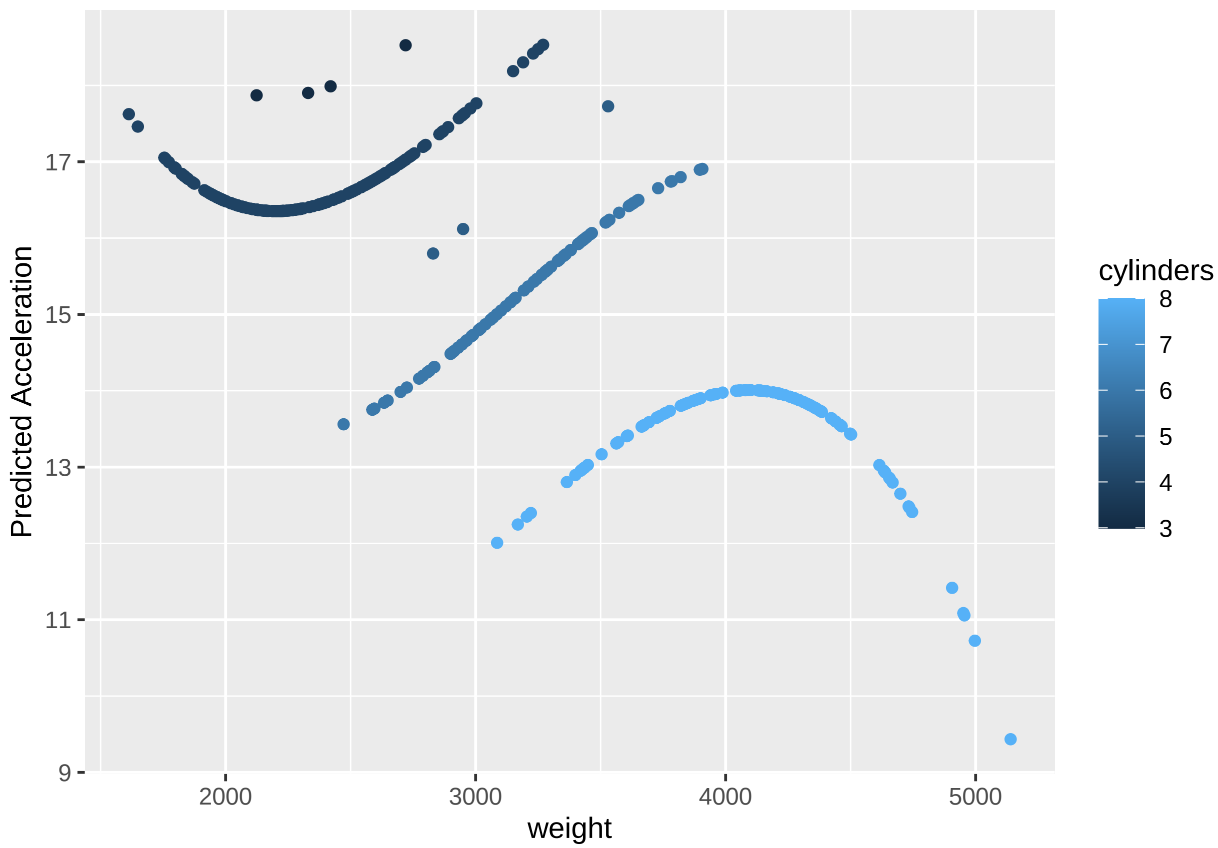

```r
# Poly models

## Predict mpg from horsepower and number of cylinders
poly.fit <- lm(mpg ~ poly(horsepower,3) + cylinders)
pred <- predict(poly.fit, newdata = Auto)

ggplot(data = Auto, aes(x = horsepower, y = pred, color = cylinders)) +
  geom_point() +
  ylab("Predicted mpg")
```




```
## predict mpg from weight and number of cylinders
poly.fit <- lm(mpg ~ poly(weight,3) + cylinders)
pred <- predict(poly.fit, newdata = Auto)

ggplot(data = Auto, aes(x = weight, y = pred, color = cylinders)) +
  geom_point() +
  ylab("Predicted mpg")
```




```r
## predict acceleration from weight and number of cylinders
poly.fit <- lm(acceleration ~ poly(weight, 3) + cylinders)
pred <- predict(poly.fit, newdata = Auto)

ggplot(data = Auto, aes(x = weight, y = pred, color = cylinders)) +
  geom_point() +
  ylab("Predicted Acceleration")
```




```
summary(poly.fit)
```

```
##
## Call:
## lm(formula = acceleration ~ poly(weight, 3) + cylinders)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.4882 -1.5422 -0.1442  1.3938  9.2486
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)        23.7412     0.9712  24.444  < 2e-16 ***
## poly(weight, 3)1  22.6254     5.8082   3.895 0.000115 ***
## poly(weight, 3)2  -5.4493     2.2968  -2.373 0.018154 *
## poly(weight, 3)3 -12.8328     2.6209  -4.896 1.44e-06 ***
## cylinders         -1.4985     0.1762  -8.504 4.05e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.297 on 387 degrees of freedom
## Multiple R-squared:  0.3141, Adjusted R-squared:  0.307
## F-statistic:  44.3 on 4 and 387 DF,  p-value: < 2.2e-16
```

Question 3 - (ISLR Ch.7 Q9)

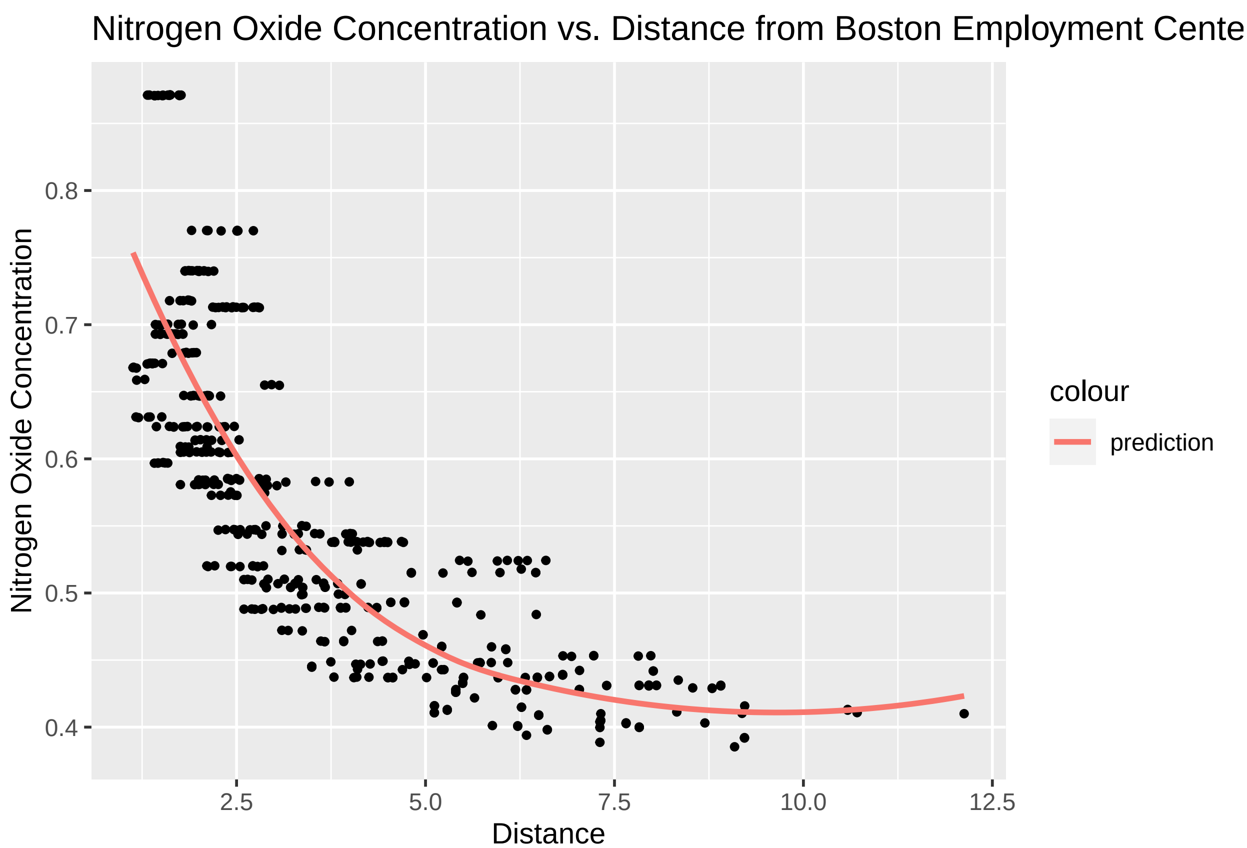
```
library(MASS)
attach(Boston)
data(Boston)
```

a - Using poly() to predict nox with dis0

```
poly.fit <- lm(nox ~ poly(dis, 3))
summary(poly.fit)
```

```
##
## Call:
## lm(formula = nox ~ poly(dis, 3))
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.121130 -0.040619 -0.009738  0.023385  0.194904
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)    0.554695   0.002759 201.021  < 2e-16 ***
## poly(dis, 3)1 -2.003096   0.062071 -32.271  < 2e-16 ***
## poly(dis, 3)2  0.856330   0.062071  13.796  < 2e-16 ***
## poly(dis, 3)3 -0.318049   0.062071  -5.124 4.27e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.06207 on 502 degrees of freedom
## Multiple R-squared:  0.7148, Adjusted R-squared:  0.7131
## F-statistic: 419.3 on 3 and 502 DF,  p-value: < 2.2e-16
```

```
poly.pred <- predict(poly.fit, newdata = Boston)
ggplot(data = Boston) +
  geom_jitter(aes(x=dis, y = nox), size = 1) +
  geom_smooth(aes(x = dis, y = poly.pred, color = "prediction"), se = F) +
  labs(title = "Nitrogen Oxide Concentration vs. Distance from Boston Employment Centers",
       x = "Distance",
       y = "Nitrogen Oxide Concentration")
```



b. Polynomial fits for a range of degrees

```
RSS.list <- c(rep(0,10))
for(i in 1:10) {
  poly.fit <- lm(nox ~ poly(dis, i))
  pred <- predict(poly.fit, newdata = Boston)
  RSS <- sum((Boston$nox - pred)^2)
  RSS.list[i] <- RSS
}

RSS.list
```

```
##  [1] 2.768563 2.035262 1.934107 1.932981 1.915290 1.878257 1.849484 1.835630
##  [9] 1.833331 1.832171
```

c. Cross validation to select degree

The following uses cross validation to determine the optimal degree polynomial to be used to fit the data. The mean squared error is used as the deciding metric. This approach was used to find that a first degree polynomial is the best fit for this data.

```
k <- 5  # Number of folds
ncv <- ceiling(nrow(Boston) / k)  # Number observations per fold
cv.ind <- rep(1:k, ncv)
cv.ind.rand <- sample(cv.ind, nrow(Boston), replace = F)
```

```r
# Vectors to track error
cv.error <- c()
MSE.cv <- c()

for(i in 1:10) {
  for(j in 1:k){  # For each fold
    dis.train <- Boston[cv.ind.rand != j, 'dis']  # Training predictor
    nox.train <- Boston[cv.ind.rand != j, 'nox']  # Training response
    poly.fit <- lm(nox.train ~ poly(dis.train, i), data = Boston)

    dis.test <- data.frame(Boston[cv.ind.rand == j, 'dis'])  # Test predictor
    colnames(dis.test) <- 'dis'
    nox.test <- Boston[cv.ind.rand == j, 'nox']  # Test response

    cv.nox.pred <- predict(poly.fit, newdata = dis.test)  # Predict response

    MSE.cv[j] <- sum((nox.test - cv.nox.pred)^2) / nrow(dis.test)  # Calculate MSE for that fold
  }
  cv.error[i] <- mean(MSE.cv)
}

# Return the lowest MSE:
(min <- which.min(cv.error))
```

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

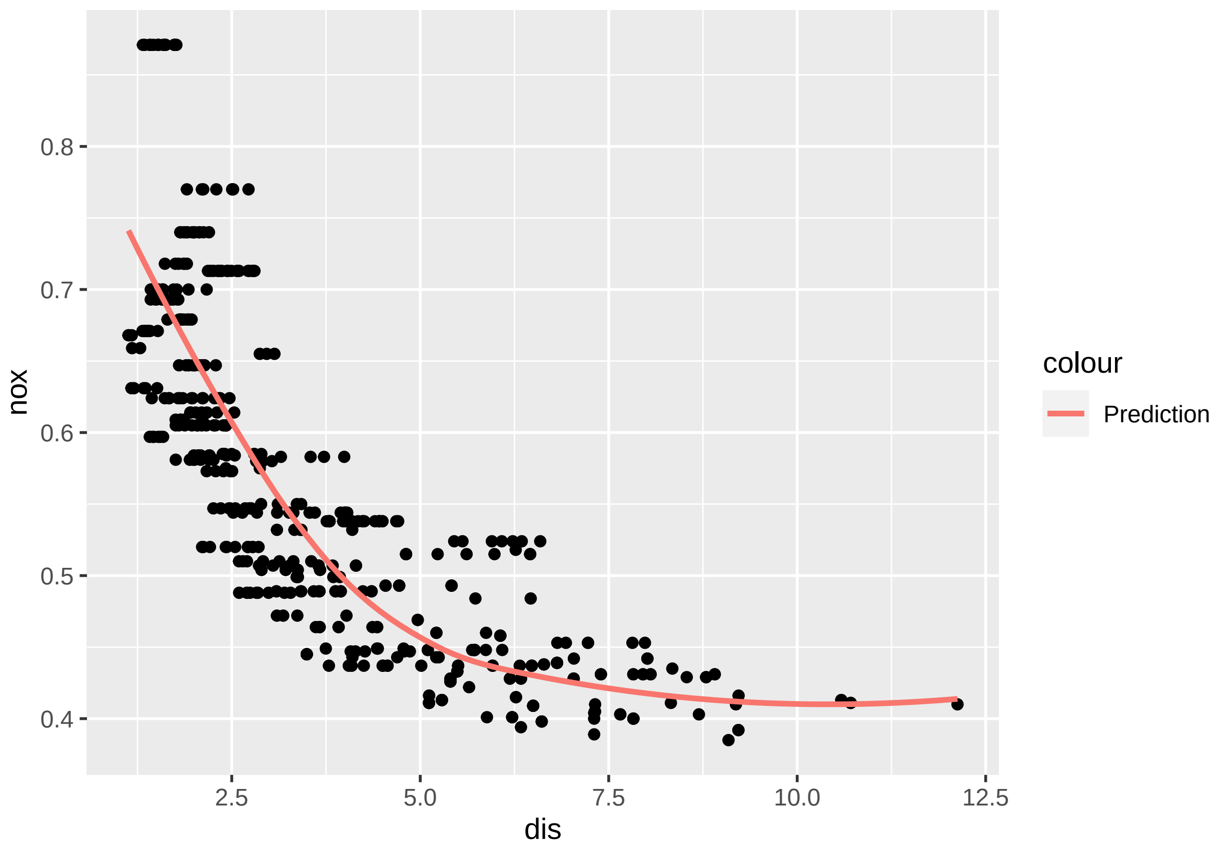
```r
cv.error
```

```
##  [1] 0.08287093 0.08579723 0.08821119 0.08834312 0.08863263 0.08969719
##  [7] 0.09057834 0.09075914 0.09083127 0.09077693
```

d. Use the bs() function to fit a regression spline

The following fits a regression spline to predict nox using dis. I chose to use 4 degrees of freedom (3 internal knots) because it seems natural to split the data into quarters.

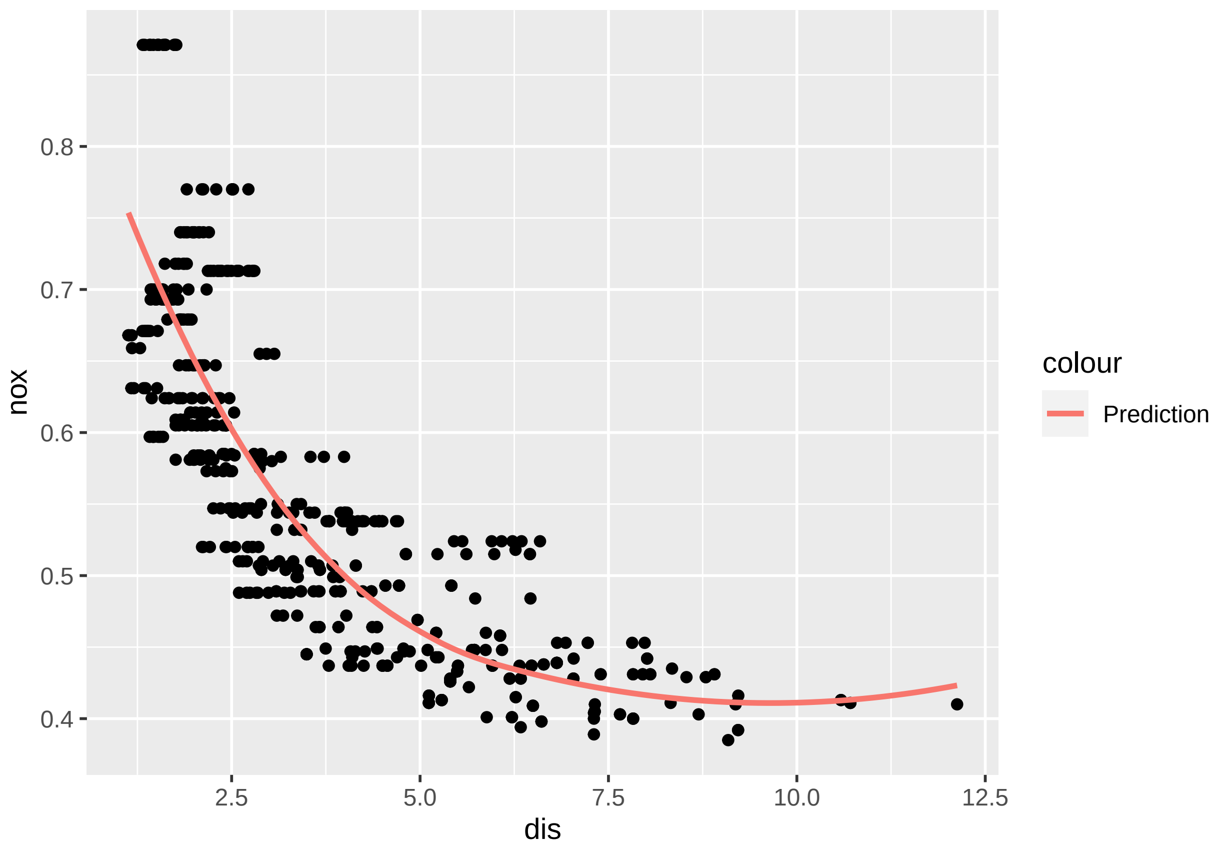
```r
spline.fit <- lm(nox ~ bs(dis, df = 4), data = Boston)
spline.pred <- predict(spline.fit, newdata = Boston)

ggplot(data = Boston) +
  geom_point(aes(x = dis, y = nox)) +
  geom_smooth(aes(x = dis, y = spline.pred, color = "Prediction"), se = F)
```

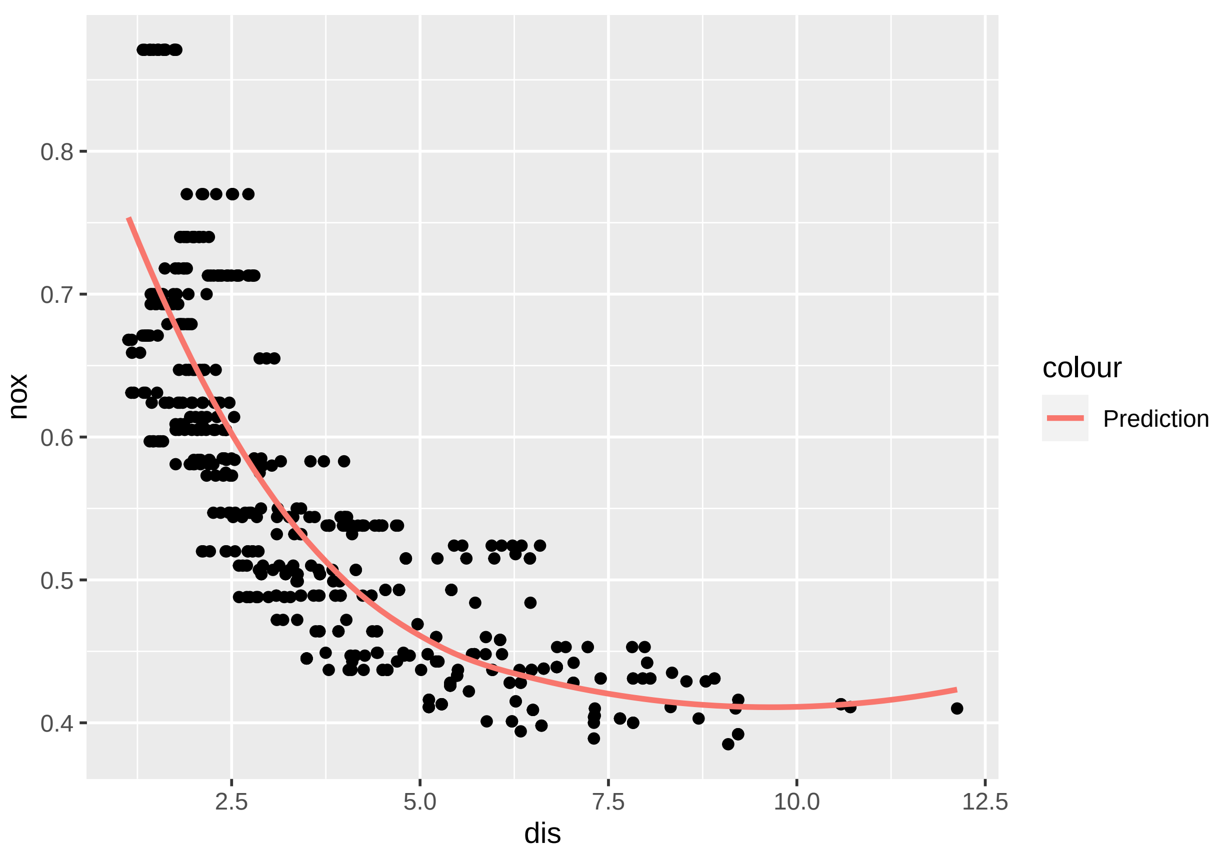


e. Fit regression splines for a range of degrees of freedom

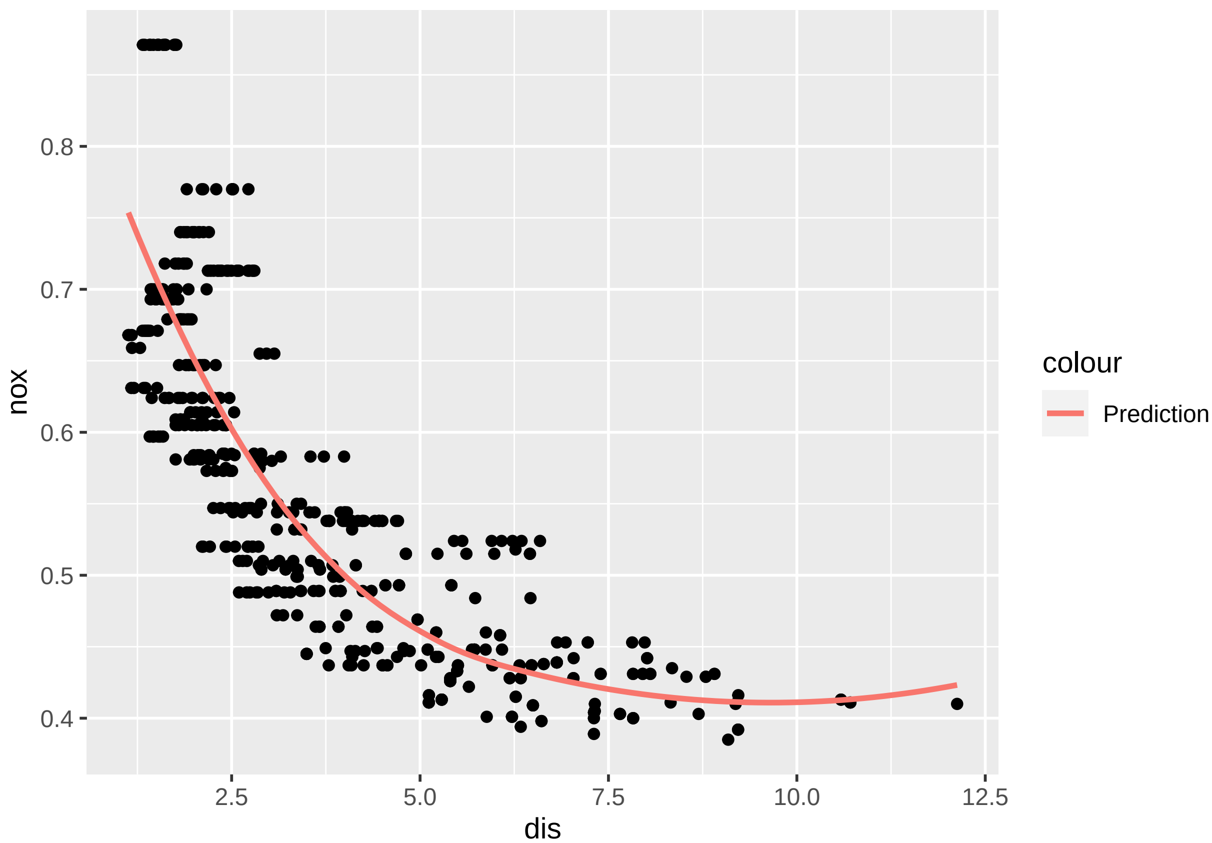
Using degrees of freedom from 1 to 5 the model that had the lowest RSS at a value of 1.84 was the most flexible model with 5 degrees of freedom.

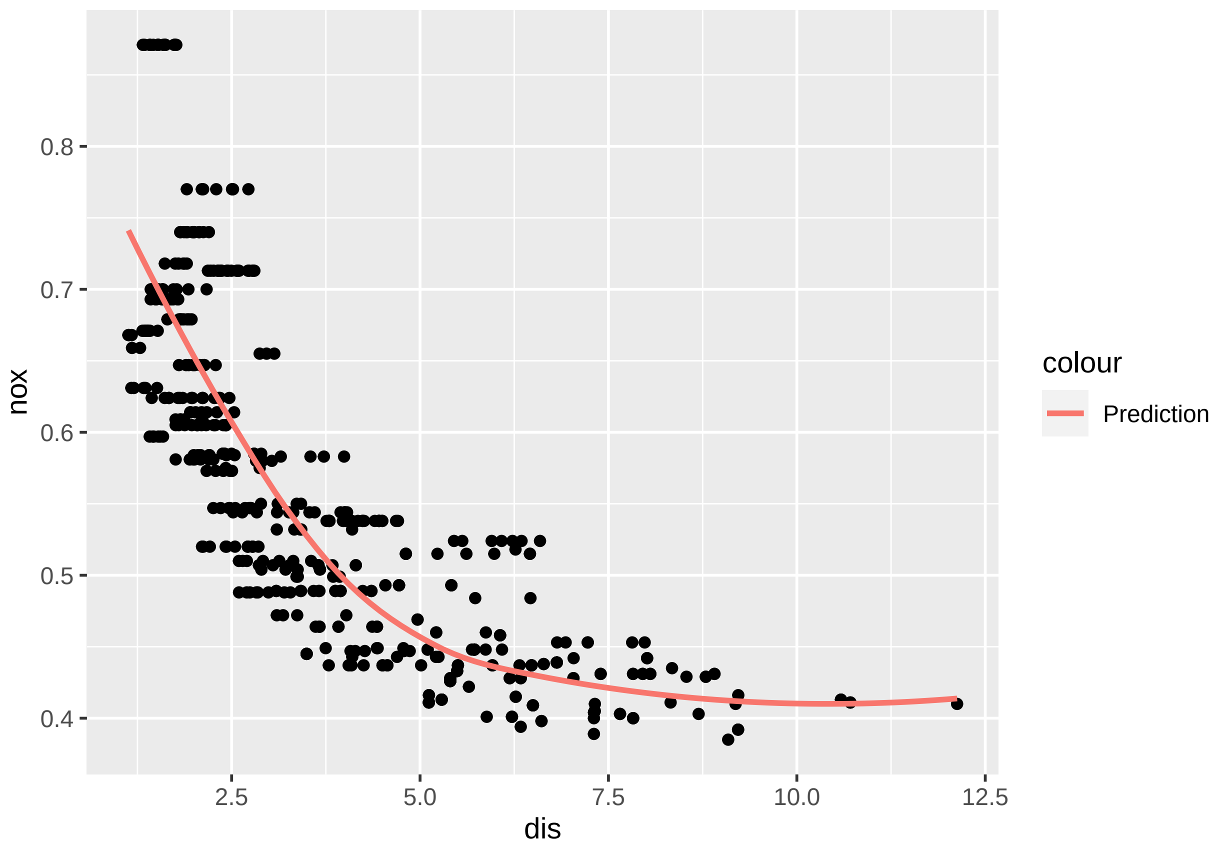
```
RSS.list <- c(rep(0,5))
plot.list <- c()
for(i in 1:5) {
  spline.fit <- lm(nox ~ bs(dis, df = i))
  spline.pred <- predict(spline.fit, newdata = Boston)
  RSS <- sum((Boston$nox - spline.pred)^2)
  RSS.list[i] <- RSS

  print(ggplot(data = Boston) +
    geom_point(aes(x = dis, y = nox)) +
    geom_smooth(aes(x = dis, y = spline.pred, color = "Prediction"), se = F))
}
```


0.8
0.7
0.6
0.5
0.4
nox
2.5
5.0
7.5
10.0
12.5
dis
colour
Prediction

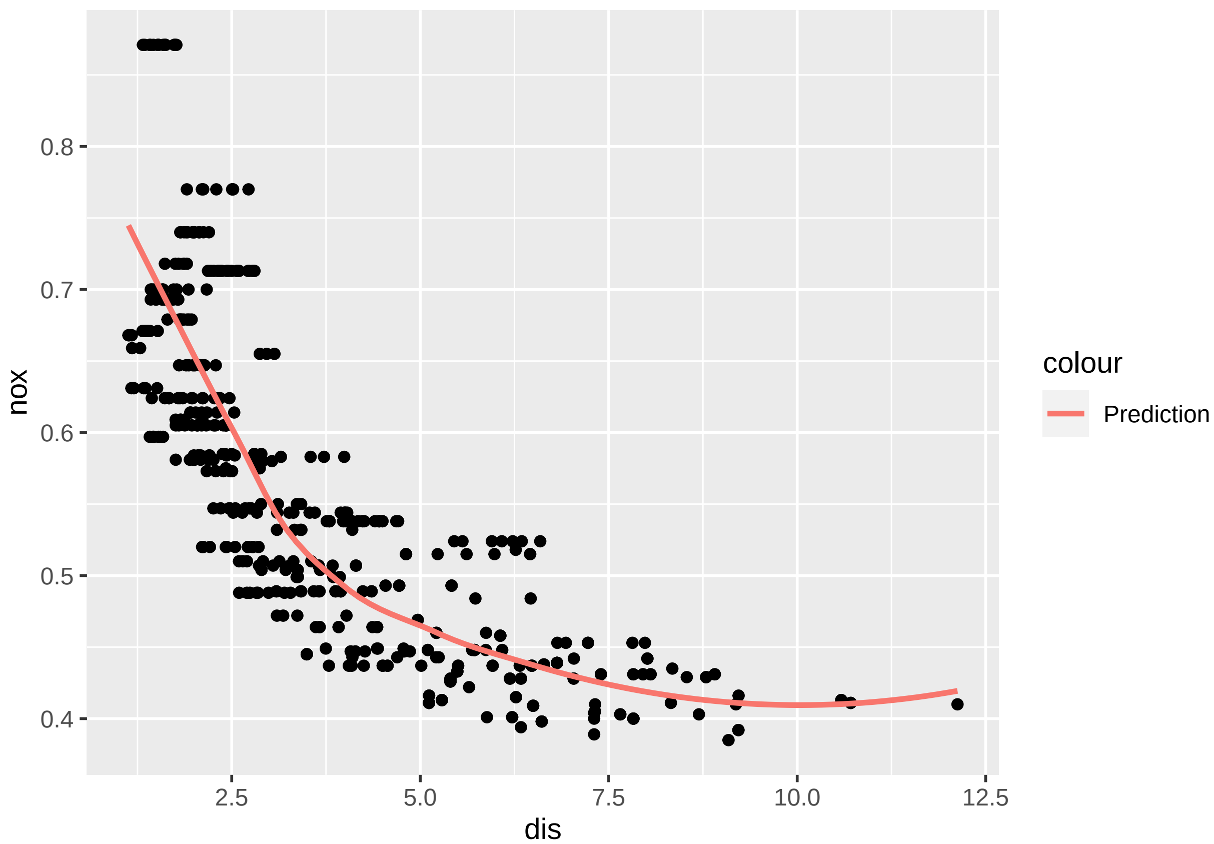


0.8
0.7
0.6
0.5
0.4
nox
2.5
5.0
7.5
10.0
12.5
dis
colour
Prediction


nox
0.8
0.7
0.6
0.5
0.4
2.5
5.0
7.5
10.0
12.5
dis
colour
Prediction



0.8
0.7
0.6
0.5
0.4
nox
2.5
5.0
7.5
10.0
12.5
dis
colour
Prediction




```
RSS.list
```

```
## [1] 1.934107 1.934107 1.934107 1.922775 1.840173
```

```
(min <- which.min(RSS.list))
```

```
## [1] 5
```

```
RSS.list[min]
```

```
## [1] 1.840173
```

f - use cross validation to select optimal degrees freedom

Using cross-validation to determine the optimal number of degrees of freedom results in 4 degrees of freedom which has an MSE of .0876.

```
k <- 5  # Number of folds
ncv <- ceiling(nrow(Boston)/k)  # Number observations per fold
cv.ind = rep(1:k, ncv)  # Used to assign folds to observations
cv.ind.rand = sample(cv.ind, nrow(Boston), replace = F)  # Randomize cv.ind

# Vectors used to track error
cv.error <- c()  # Avg MSE between folds for a given knot
MSE.cv <- c()  # MSE for a given fold


for(i in 1:5){  # For every knot in the list
  for(j in 1:k){  # For each fold
```

```
    dis.train <- Boston[cv.ind.rand != j, 'dis']  # Training predictor
    nox.train <- Boston[cv.ind.rand != j, 'nox']  # Training response
    model <- lm(nox.train ~ bs(dis.train, df = i), data=Boston)

    dis.test <- data.frame(Boston[cv.ind.rand == j, 'dis'])  # Test predictor
    colnames(dis.test) <- 'dis'
    nox.test <- Boston[cv.ind.rand == j, 'nox']  # Test response

    cv.nox.pred <- predict(model, newdata=data.frame())  # Predict response

    MSE.cv[j] <- sum((nox.test - cv.nox.pred)^2) / nrow(dis.test)  # Calculate MSE for that fold
  }
  cv.error[i] <- mean(MSE.cv)
}

# Return the lowest MSE:
(min <- which.min(cv.error))
```

```
## [1] 4
```

```
cv.error[min]
```

```
## [1] 0.08784664
```

```
cv.error
```

```
## [1] 0.08788883 0.08788883 0.08788883 0.08784664 0.09025546
```