HW 7

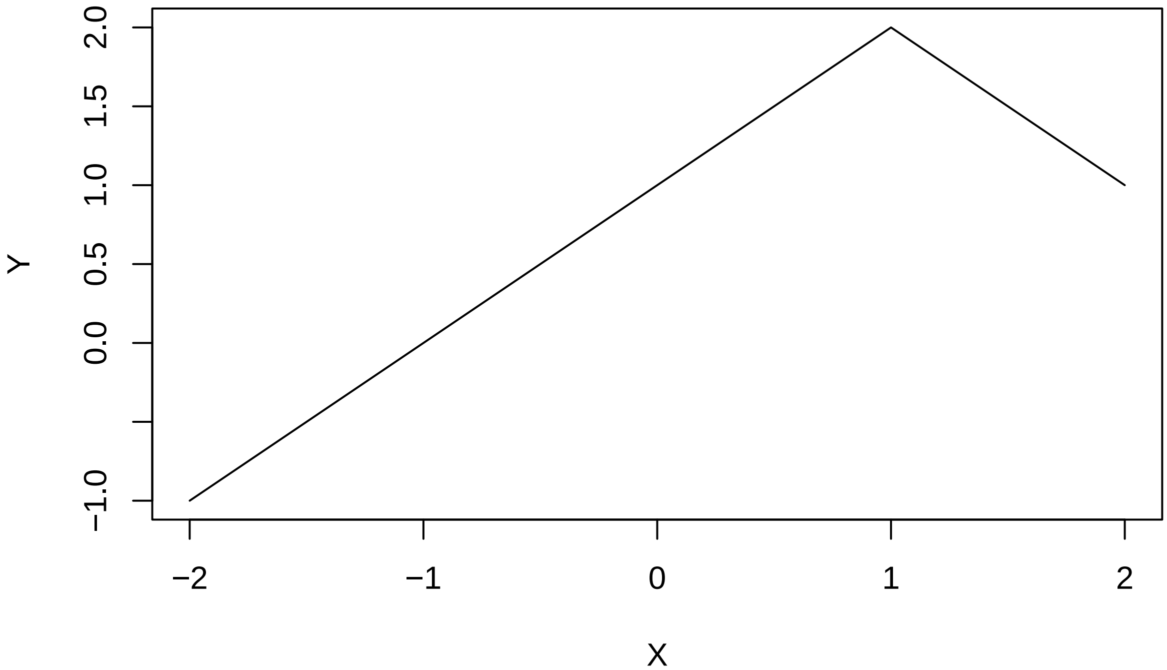
Jared Andreatta (According to Canvas I'm the only one in my group?)

2025-04-27

Problem 3

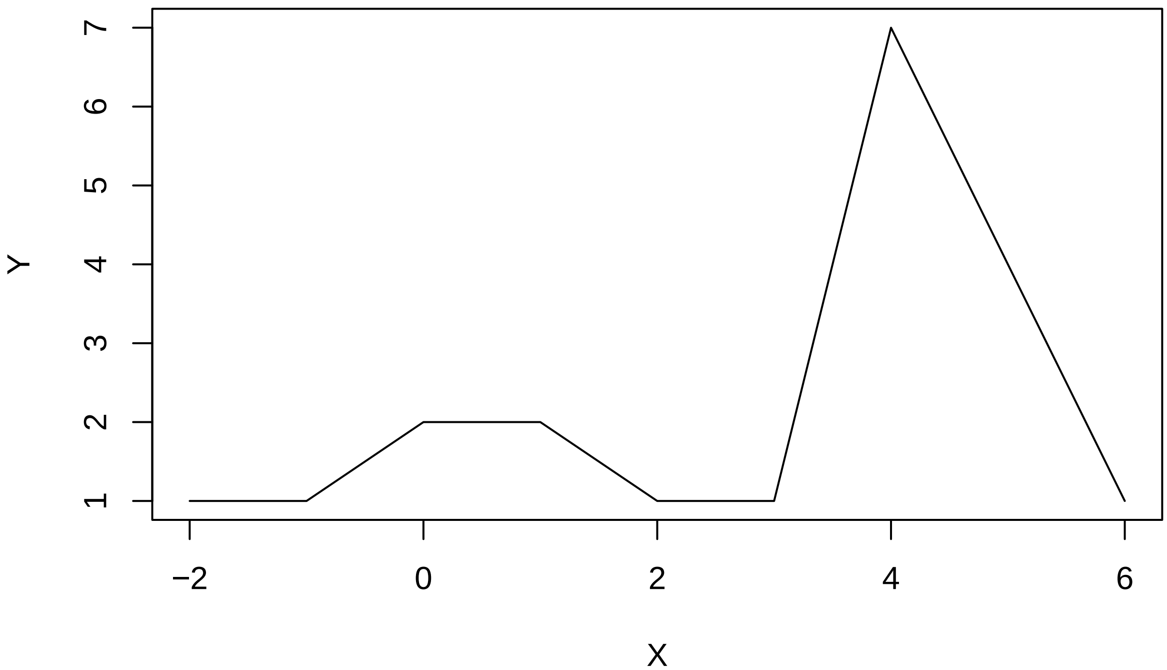
```
beta_1 <- 1
beta_0 <- 1
beta_2 <- -2
X <- seq(from=-2,to=2,by=1)

b_1 <- X
b_2 <- (X-1)^2 * ifelse(X >= 1, 1, 0)

Y = beta_0 + beta_1 * b_1 + beta_2 *b_2

plot(X,Y,type = "line")
```

```
## Warning in plot.xy(xy, type, ...): plot type 'line' will be truncated to first
## character
```



Problem 4

```r
X <- seq(-2, 6, by = 1)

b_1 <- ifelse(0 <= X & X <= 2, 1, 0) - (X - 1) * ifelse(1 <= X & X <= 2, 1, 0)
b_2 <- (X - 3) * ifelse(3 <= X & X <= 4, 1, 0) + ifelse(4 <= X & X <= 5, 1, 0)

b <- cbind(1, b_1, b_2)
beta <- c(1, 1, 3)

Y <- b %*% beta

plot(X, Y, type = "l")
```



Problem 6

a.

According to CV errors, the optimal degree is 6. However, according to the ANOVA table, the optimal level is 4, which is indicated by the insignificant p-values after d=4.

```r
library(ISLR2)
library(ggplot2)
library(boot)
set.seed(11)

### CV ###
cv.errors <- numeric(10)

for (d in 1:10){
  fit <- glm(wage~poly(age,d,raw=TRUE), data=Wage)
  cv.errors[d] <- cv.glm(Wage, fit, K=10)$delta[1]
```

```r
}

opt.degree <- which.min(cv.errors)
opt.degree
```

```
## [1] 6
```

```r
print(cv.errors)
```

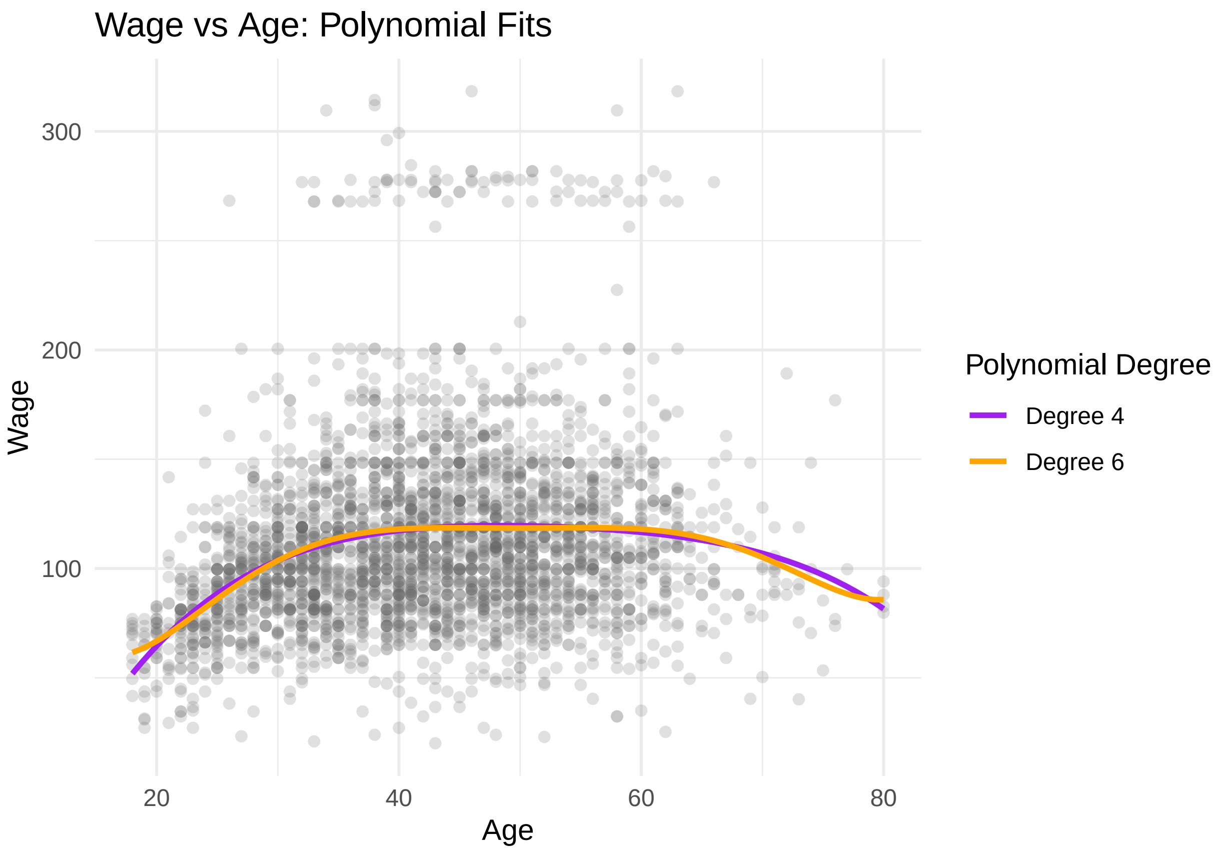
```
##  [1] 1676.122 1599.190 1596.520 1594.568 1594.525 1593.775 1595.154 1595.313
##  [9] 1596.254 1598.146
```

```r
### ANOVA ###
anova.fits <- lapply(1:10, function(i) lm(wage ~ poly(age, i, raw=TRUE), data = Wage))
anova.table <- do.call(anova, anova.fits)
print(anova.table)
```

```
## Analysis of Variance Table
##
## Model  1: wage ~ poly(age, i, raw = TRUE)
## Model  2: wage ~ poly(age, i, raw = TRUE)
## Model  3: wage ~ poly(age, i, raw = TRUE)
## Model  4: wage ~ poly(age, i, raw = TRUE)
## Model  5: wage ~ poly(age, i, raw = TRUE)
## Model  6: wage ~ poly(age, i, raw = TRUE)
## Model  7: wage ~ poly(age, i, raw = TRUE)
## Model  8: wage ~ poly(age, i, raw = TRUE)
## Model  9: wage ~ poly(age, i, raw = TRUE)
## Model 10: wage ~ poly(age, i, raw = TRUE)
##    Res.Df     RSS Df Sum of Sq        F    Pr(>F)
## 1    2998 5022216
## 2    2997 4793430  1    228786 143.7638 < 2.2e-16 ***
## 3    2996 4777674  1     15756   9.9005  0.001669 **
## 4    2995 4771604  1      6070   3.8143  0.050909 .
## 5    2994 4770322  1      1283   0.8059  0.369398
## 6    2993 4766389  1      3932   2.4709  0.116074
## 7    2992 4763834  1      2555   1.6057  0.205199
## 8    2991 4763707  1       127   0.0796  0.777865
## 9    2990 4756703  1      7004   4.4014  0.035994 *
## 10   2989 4756701  1         3   0.0017  0.967529
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
### Plotting ###
ggplot(Wage, aes(age, wage)) +
  geom_point(alpha = 0.2, color = "grey40") +
  stat_smooth(aes(color = "Degree 4"),
              method  = "lm",
              formula = y ~ poly(x, 4),
              se      = FALSE,
              size    = 1) +

  stat_smooth(aes(color = "Degree 6"),
              method  = "lm",
              formula = y ~ poly(x, 6),
              se      = FALSE,
```

```r
            size    = 1) +

  scale_color_manual(name   = "Polynomial Degree",
                     values = c("Degree 4" = "purple",
                                "Degree 6" = "orange")) +

  labs(title = "Wage vs Age: Polynomial Fits",
       x     = "Age",
       y     = "Wage") +
  theme_minimal()
```



b.

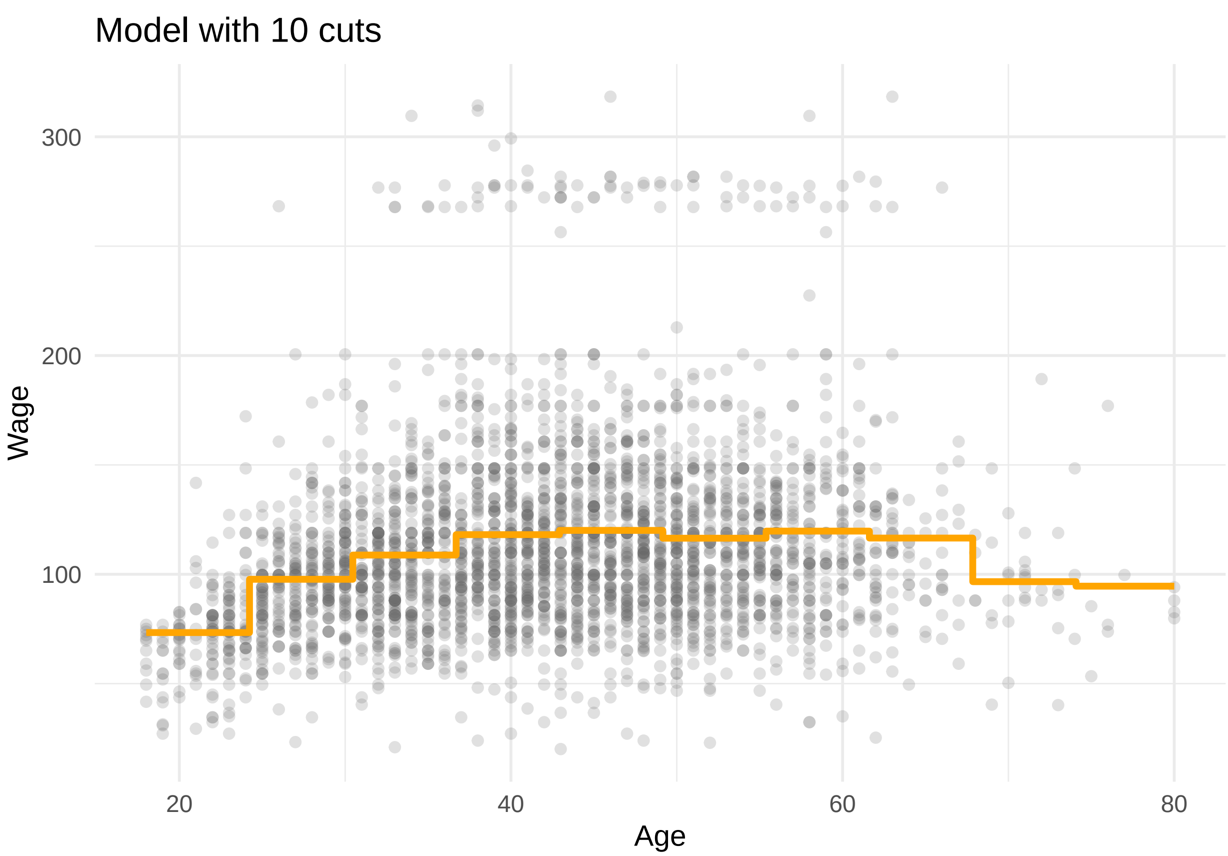
```r
set.seed(123)

### CV for number of cuts ###
cv.errors <- numeric(10)
for (i in 1:10) {
  Wage$cut <- cut(Wage$age, i+1)
  fit_i <- glm(wage ~ cut, data = Wage)
  cv.errors[i] <- cv.glm(Wage, fit_i, K = 10)$delta[1]
}

opt.i <- which.min(cv.errors)
opt.i
```

```
## [1] 10
```

```r
print(cv.errors[1:10])
```

```
##  [1] 1734.784 1683.196 1636.622 1631.955 1621.842 1612.172 1603.495 1607.430
##  [9] 1605.066 1599.293
```

```r
final_fit <- glm(wage ~ cut(age, opt.i),
                 data = Wage)

### Plotting with optimal model (i=10) ###
x.grid <- seq(min(Wage$age), max(Wage$age), length.out = 200)

preds <- predict(final_fit, newdata = data.frame(age = x.grid))
pred_df <- data.frame(age=x.grid, wage=preds)

ggplot() +
  geom_point(data = Wage,
             aes(age, wage),
             alpha = 0.2,
             color = "grey40") +
  geom_step(data = pred_df,
            aes(age, wage),
            color = "orange",
            size  = 1.2) +
  labs(title = "Model with 10 cuts",
       x = "Age", y = "Wage") +
  theme_minimal()
```



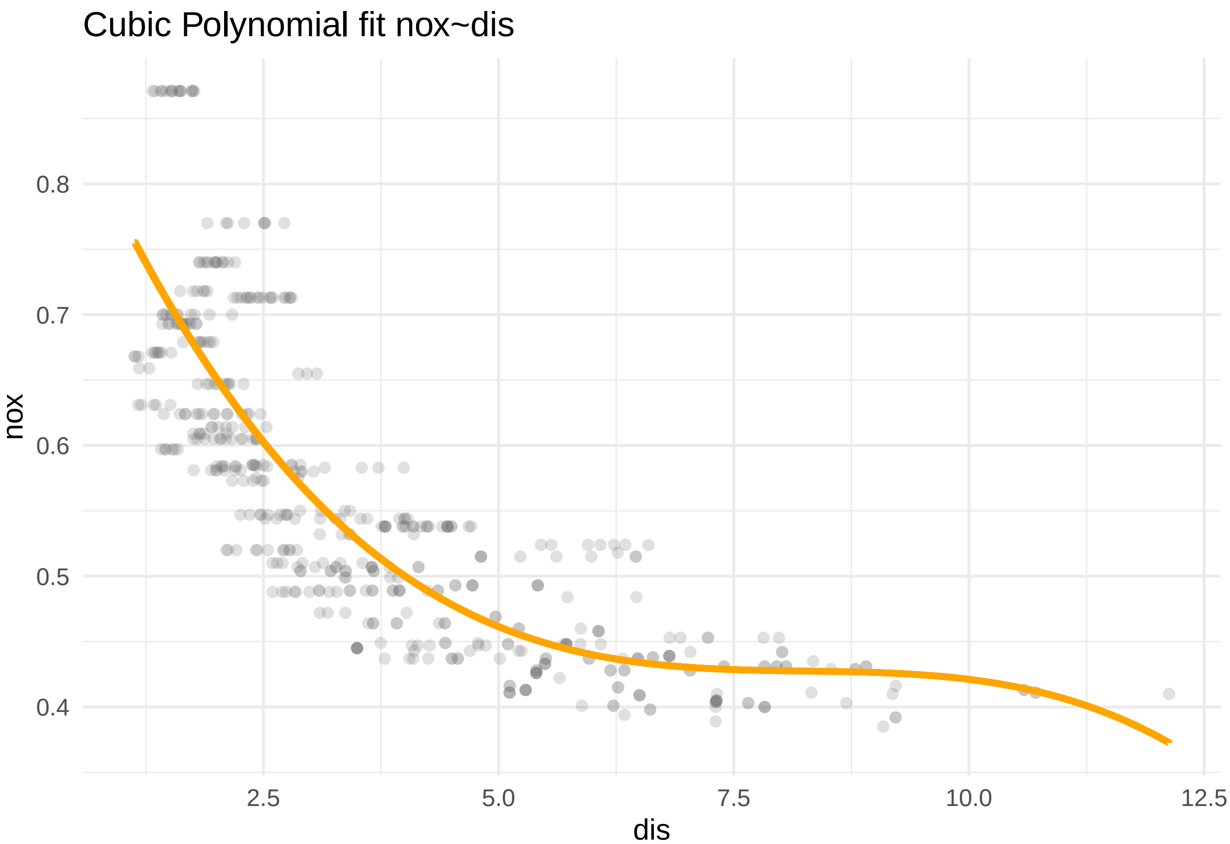
Problem 9

a.

```r
### Fitting ###
fit <- lm(nox~poly(dis, 3, raw=TRUE), data=Boston)
summary <- summary(fit)

summary
```

```
##
## Call:
## lm(formula = nox ~ poly(dis, 3, raw = TRUE), data = Boston)
##
## 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)               0.9341281  0.0207076  45.110  < 2e-16 ***
## poly(dis, 3, raw = TRUE)1 -0.1820817  0.0146973 -12.389  < 2e-16 ***
## poly(dis, 3, raw = TRUE)2  0.0219277  0.0029329   7.476 3.43e-13 ***
## poly(dis, 3, raw = TRUE)3 -0.0008850  0.0001727  -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
```

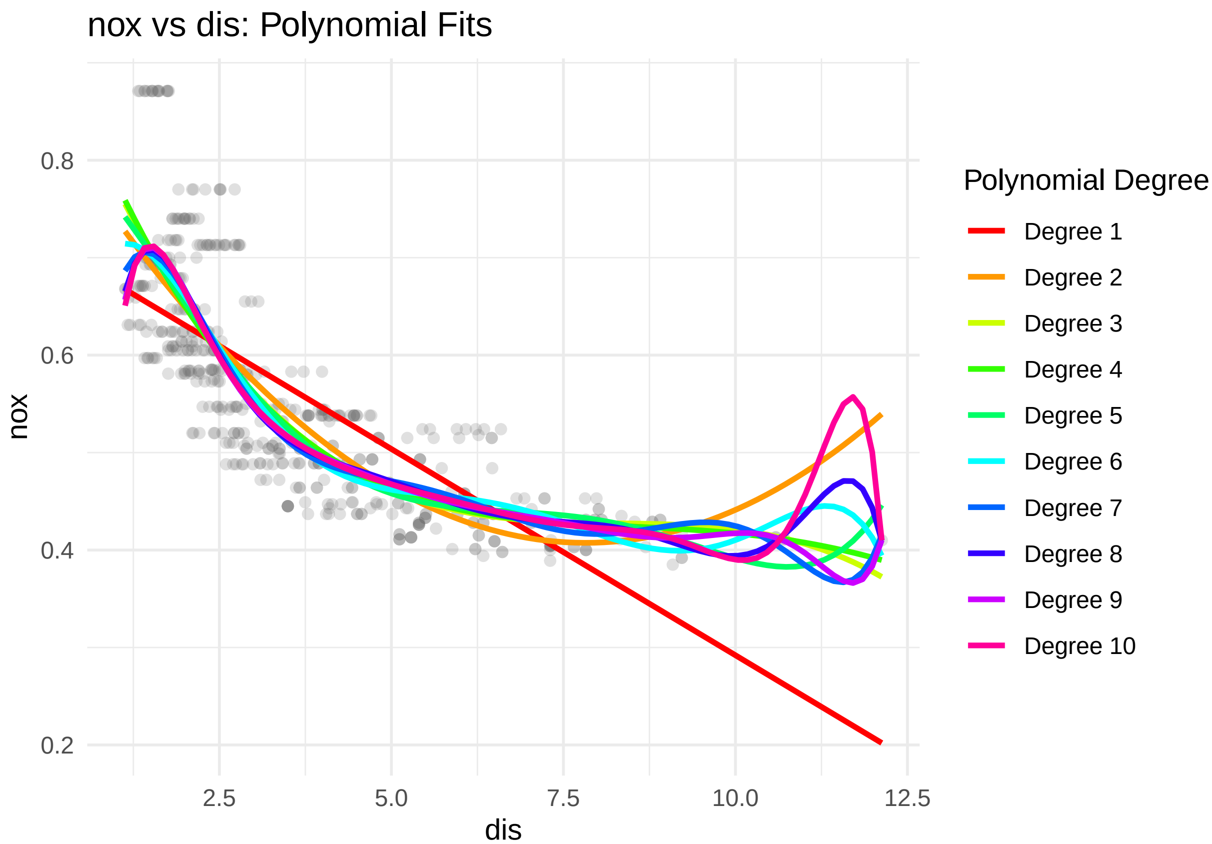
```r
### Plotting ###
x.grid <- seq(min(Boston$dis), max(Boston$dis), length.out = 1000)
preds <- predict(fit, newdata = data.frame(dis=x.grid))
pred_df <- data.frame(dis=x.grid, nox=preds)

ggplot() +
  geom_point(data = Boston,
             aes(dis, nox),
             alpha = 0.2,
             color = "grey40") +
  geom_step(data = pred_df,
            aes(dis, nox),
            color = "orange",
            size  = 1.2) +
  labs(title = "Cubic Polynomial fit nox~dis",
       x = "dis", y = "nox") +
  theme_minimal()
```



b.

```
### Plotting poly fits ###
ggplot(Boston, aes(dis, nox)) +
  geom_point(alpha = 0.2, color = "grey40") +

  stat_smooth(aes(color = "Degree 1"),
              method  = "lm",
              formula = y ~ poly(x, 1),
              se      = FALSE,
              size    = 1) +
  stat_smooth(aes(color = "Degree 2"),
              method  = "lm",
              formula = y ~ poly(x, 2),
              se      = FALSE,
              size    = 1) +
  stat_smooth(aes(color = "Degree 3"),
              method  = "lm",
              formula = y ~ poly(x, 3),
              se      = FALSE,
              size    = 1) +
  stat_smooth(aes(color = "Degree 4"),
              method  = "lm",
              formula = y ~ poly(x, 4),
              se      = FALSE,
              size    = 1) +
  stat_smooth(aes(color = "Degree 5"),
              method  = "lm",
```

```r
              formula = y ~ poly(x, 5),
              se      = FALSE,
              size    = 1) +
stat_smooth(aes(color = "Degree 6"),
              method  = "lm",
              formula = y ~ poly(x, 6),
              se      = FALSE,
              size    = 1) +
stat_smooth(aes(color = "Degree 7"),
              method  = "lm",
              formula = y ~ poly(x, 7),
              se      = FALSE,
              size    = 1) +
stat_smooth(aes(color = "Degree 8"),
              method  = "lm",
              formula = y ~ poly(x, 8),
              se      = FALSE,
              size    = 1) +
stat_smooth(aes(color = "Degree 9"),
              method  = "lm",
              formula = y ~ poly(x, 9),
              se      = FALSE,
              size    = 1) +
stat_smooth(aes(color = "Degree 10"),
              method  = "lm",
              formula = y ~ poly(x, 10),
              se      = FALSE,
              size    = 1) +

scale_color_manual(
  name   = "Polynomial Degree",
  values = setNames(rainbow(10), paste0("Degree ", 1:10)),
  breaks = paste0("Degree ", 1:10)
) +

labs(title = "nox vs dis: Polynomial Fits",
     x     = "dis",
     y     = "nox") +
theme_minimal()
```



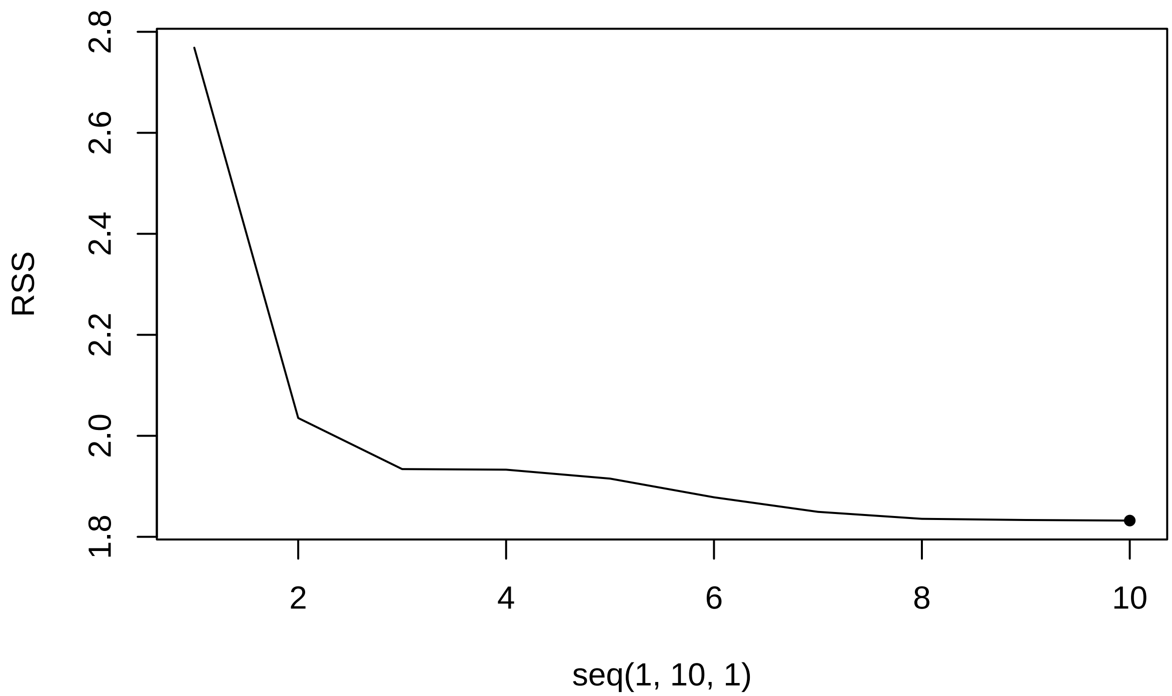

```r
### RSS for poly fits ###
RSS=numeric(10)
for(i in 1:10){
  fit=lm(nox~poly(dis,i,raw=TRUE), data=Boston)
  preds=predict(fit,Boston)
  RSS[i]=sum((preds-Boston$nox)^2)
}

RSS
```

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

```r
### Plotting RSS ###
min.RSS <- which.min(RSS)
min.RSS
```

```
## [1] 10
```

```r
plot(seq(1,10,1), RSS, type="l")
points(10, RSS[10], pch=20)
```



c.

I'm a bigger proponent for using ANOVA in this case. Since CV errors in the cv.glm function seek to minimize MSE, the MSE will decrease montonically with model complexity, so the CV method chooses d=10. On the other hand, we can see a dip in significance of adding another degree past d=3 in the ANOVA table. This indicates that the extra model complexity is not necessary and is likely fit well enough by a cubic polynomial. This is advantageous in model parsimony and reducing overfitting.

```
set.seed(10101)

cv.errors <- numeric(10)

for (i in 1:10){
  fit <- glm(nox~poly(dis,i,raw=TRUE), data=Boston)
  cv.errors[i] <- cv.glm(Boston,fit,K=10)$delta[1]
}

cv.errors
```

```
##  [1] 0.005508893 0.004073774 0.003861801 0.003888012 0.004116645 0.005773165
##  [7] 0.013607922 0.014419665 0.010810368 0.003713456
```

```
which.min(cv.errors)
```

```
## [1] 10
```

```
### ANOVA ###
anova.fits <- lapply(1:10, function(i) lm(nox ~ poly(dis, i,raw=TRUE), data = Boston))
anova.table <- do.call(anova, anova.fits)
print(anova.table)
```

```
## Analysis of Variance Table
##
## Model  1: nox ~ poly(dis, i, raw = TRUE)
```

```
## Model  2: nox ~ poly(dis, i, raw = TRUE)
## Model  3: nox ~ poly(dis, i, raw = TRUE)
## Model  4: nox ~ poly(dis, i, raw = TRUE)
## Model  5: nox ~ poly(dis, i, raw = TRUE)
## Model  6: nox ~ poly(dis, i, raw = TRUE)
## Model  7: nox ~ poly(dis, i, raw = TRUE)
## Model  8: nox ~ poly(dis, i, raw = TRUE)
## Model  9: nox ~ poly(dis, i, raw = TRUE)
## Model 10: nox ~ poly(dis, i, raw = TRUE)
##    Res.Df    RSS Df Sum of Sq        F    Pr(>F)
## 1     504 2.7686
## 2     503 2.0353  1   0.73330 198.1169 < 2.2e-16 ***
## 3     502 1.9341  1   0.10116  27.3292 2.535e-07 ***
## 4     501 1.9330  1   0.00113   0.3040  0.581606
## 5     500 1.9153  1   0.01769   4.7797  0.029265 *
## 6     499 1.8783  1   0.03703  10.0052  0.001657 **
## 7     498 1.8495  1   0.02877   7.7738  0.005505 **
## 8     497 1.8356  1   0.01385   3.7429  0.053601 .
## 9     496 1.8333  1   0.00230   0.6211  0.431019
## 10    495 1.8322  1   0.00116   0.3133  0.575908
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```