DSI_06- HW6 pg 324

Julia Gallucci

2023-03-05

9. This question uses the variables dis (the weighted mean of distances to five Boston employment centers) and nox (nitrogen oxides concentration in parts per 10 million) from the Boston data. We will treat dis as the predictor and nox as the response.

```
#install.packages("ISLR2")  Install package if you haven't already
library(ISLR2) #load library
df<- Boston #save boston dataset as a variable "df"
head(df)
```

```
##      crim zn indus chas   nox    rm  age    dis rad tax ptratio lstat medv
## 1 0.00632 18  2.31    0 0.538 6.575 65.2 4.0900   1 296    15.3  4.98 24.0
## 2 0.02731  0  7.07    0 0.469 6.421 78.9 4.9671   2 242    17.8  9.14 21.6
## 3 0.02729  0  7.07    0 0.469 7.185 61.1 4.9671   2 242    17.8  4.03 34.7
## 4 0.03237  0  2.18    0 0.458 6.998 45.8 6.0622   3 222    18.7  2.94 33.4
## 5 0.06905  0  2.18    0 0.458 7.147 54.2 6.0622   3 222    18.7  5.33 36.2
## 6 0.02985  0  2.18    0 0.458 6.430 58.7 6.0622   3 222    18.7  5.21 28.7
```

(a) Use the poly() function to fit a cubic polynomial regression to predict nox using dis. Report the regression output, and plot the resulting data and polynomial fits.

```
poly.fit <- lm(nox ~ poly(dis,3), data = df) #fit a cubic polynomial regression
summary(poly.fit) #summmary of output
```

```
##
## Call:
## lm(formula = nox ~ poly(dis, 3), data = df)
##
## 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 ***
## ---
```

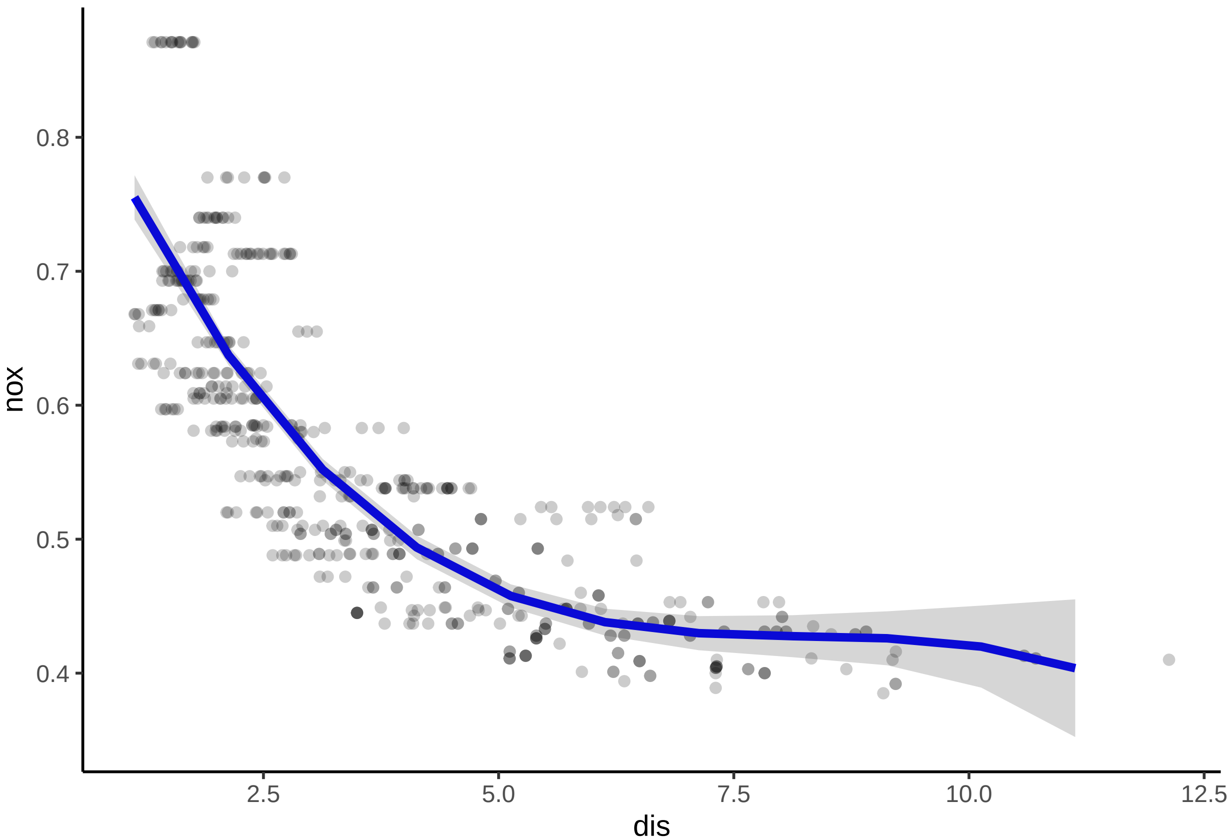
```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.06207 on 502 degrees of freedom
## Multiple R-squared:  0.7148, Adjusted R-squared:  0.7131
## F-statistic: 419.3 on 3 and 502 DF,  p-value: < 2.2e-16
```

The model finds each power (1,2, and 3) of the dis coefficient to be statistically significant (p<0.05).

Plot resulting data and cubic polynomial fit

```
dis.range <- seq(from = min(df$dis), to = max(df$dis))
pred <-  predict(poly.fit, newdata = list(dis = dis.range), se = TRUE) #assign predicted response to pr
conf.int <- cbind(pred$fit + 2 * pred$se.fit, pred$fit - 2 * pred$se.fit)
predictions <- data.frame(DIS = dis.range, NOX = pred$fit,upper = conf.int[, 1], lower = conf.int[, 2])
#install.packages('ggplot2') install ggplot2 if you havent already done so
library(ggplot2)
ggplot() +
    geom_point(data= df, aes(dis, nox), col = 'Black', alpha = 0.2) + #points representing observed val
    geom_line(data = predictions, aes(DIS, NOX), size = 1.5, col = 'Blue') + #line representing predict
    geom_ribbon(data = predictions, aes(x = DIS, ymin = lower, ymax = upper), alpha = 0.2) + theme_clas
```

```
## Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use `linewidth` instead.
```



The fitted line seems to describe the data well without overfitting the data. CI gets larger towards larger dis numbers with minimal datapoints

(b) Plot the polynomial fits for a range of different polynomial degrees (say, from 1 to 10), and report the associated residual sum of squares.

```
range_poly = 1:10
for (degrees in range_poly){
  poly.fit <- lm(nox ~ poly(dis,degrees), data = df) #fit a cubic polynomial regression
  dis.range <- seq(from = min(df$dis), to = max(df$dis)) #new data made for dis
  pred <-  predict(poly.fit, newdata = list(dis = dis.range), se = TRUE) #assign predicted response to p
  conf.int <- cbind(pred$fit + 2 * pred$se.fit, pred$fit - 2 * pred$se.fit) #confidence interval
  predictions <- data.frame(DIS = dis.range, NOX = pred$fit,upper = conf.int[, 1], lower = conf.int[, 2]
  cat("Summary of Model fit for ",degrees, "degrees polynomial", "\n") #give each output a title
  print(summary(poly.fit)) # print summary of output
  errors <- sum(poly.fit$residuals ^2) / (nrow(df) - ncol(df))
  cat("Residual sum of squares", errors, "\n")
  cat("--------------------------------------------------------------- \n") #print a line to separate ou
}
```

```
## Summary of Model fit for  1 degrees polynomial
##
## Call:
## lm(formula = nox ~ poly(dis, degrees), data = df)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.12239 -0.05212 -0.01257  0.04391  0.23041
##
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)         0.554695   0.003295  168.35   <2e-16 ***
## poly(dis, degrees) -2.003096   0.074116  -27.03   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.07412 on 504 degrees of freedom
## Multiple R-squared:  0.5917, Adjusted R-squared:  0.5909
## F-statistic: 730.4 on 1 and 504 DF,  p-value: < 2.2e-16
##
## Residual sum of squares 0.005615746
## ---------------------------------------------------------------
## Summary of Model fit for  2 degrees polynomial
##
## Call:
## lm(formula = nox ~ poly(dis, degrees), data = df)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.129559 -0.044514 -0.007753  0.025778  0.201882
##
## Coefficients:
##                      Estimate Std. Error t value Pr(>|t|)
## (Intercept)          0.554695   0.002828  196.16   <2e-16 ***
## poly(dis, degrees)1 -2.003096   0.063610  -31.49   <2e-16 ***
## poly(dis, degrees)2  0.856330   0.063610   13.46   <2e-16 ***
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.06361 on 503 degrees of freedom
## Multiple R-squared:  0.6999, Adjusted R-squared:  0.6987
## F-statistic: 586.4 on 2 and 503 DF,  p-value: < 2.2e-16
##
## Residual sum of squares 0.00412832
## ----------------------------------------------------------------
## Summary of Model fit for  3 degrees polynomial
##
## Call:
## lm(formula = nox ~ poly(dis, degrees), data = df)
##
## 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, degrees)1 -2.003096   0.062071 -32.271  < 2e-16 ***
## poly(dis, degrees)2  0.856330   0.062071  13.796  < 2e-16 ***
## poly(dis, degrees)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
##
## Residual sum of squares 0.003923137
## ----------------------------------------------------------------
## Summary of Model fit for  4 degrees polynomial
##
## Call:
## lm(formula = nox ~ poly(dis, degrees), data = df)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.12295 -0.04089 -0.01073  0.02290  0.19471
##
## Coefficients:
##                      Estimate Std. Error t value Pr(>|t|)
## (Intercept)          0.554695   0.002761  200.88  < 2e-16 ***
## poly(dis, degrees)1 -2.003096   0.062115  -32.25  < 2e-16 ***
## poly(dis, degrees)2  0.856330   0.062115   13.79  < 2e-16 ***
## poly(dis, degrees)3 -0.318049   0.062115   -5.12 4.36e-07 ***
## poly(dis, degrees)4  0.033547   0.062115    0.54    0.589
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.06211 on 501 degrees of freedom
## Multiple R-squared:  0.7149, Adjusted R-squared:  0.7127
## F-statistic: 314.1 on 4 and 501 DF,  p-value: < 2.2e-16
```

```
##
## Residual sum of squares 0.003920855
## ------------------------------------------------------------
## Summary of Model fit for  5 degrees polynomial
##
## Call:
## lm(formula = nox ~ poly(dis, degrees), data = df)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.123244 -0.040417 -0.008737  0.024004  0.193135
##
## Coefficients:
##                      Estimate Std. Error t value Pr(>|t|)
## (Intercept)          0.554695   0.002751 201.603  < 2e-16 ***
## poly(dis, degrees)1 -2.003096   0.061892 -32.365  < 2e-16 ***
## poly(dis, degrees)2  0.856330   0.061892  13.836  < 2e-16 ***
## poly(dis, degrees)3 -0.318049   0.061892  -5.139 3.97e-07 ***
## poly(dis, degrees)4  0.033547   0.061892   0.542   0.5880
## poly(dis, degrees)5  0.133009   0.061892   2.149   0.0321 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.06189 on 500 degrees of freedom
## Multiple R-squared:  0.7175, Adjusted R-squared:  0.7147
## F-statistic:   254 on 5 and 500 DF,  p-value: < 2.2e-16
##
## Residual sum of squares 0.003884969
## ------------------------------------------------------------
## Summary of Model fit for  6 degrees polynomial
##
## Call:
## lm(formula = nox ~ poly(dis, degrees), data = df)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.12946 -0.03989 -0.01006  0.02729  0.18797
##
## Coefficients:
##                      Estimate Std. Error t value Pr(>|t|)
## (Intercept)          0.554695   0.002727 203.377  < 2e-16 ***
## poly(dis, degrees)1 -2.003096   0.061352 -32.649  < 2e-16 ***
## poly(dis, degrees)2  0.856330   0.061352  13.958  < 2e-16 ***
## poly(dis, degrees)3 -0.318049   0.061352  -5.184 3.16e-07 ***
## poly(dis, degrees)4  0.033547   0.061352   0.547  0.58477
## poly(dis, degrees)5  0.133009   0.061352   2.168  0.03063 *
## poly(dis, degrees)6 -0.192439   0.061352  -3.137  0.00181 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.06135 on 499 degrees of freedom
## Multiple R-squared:  0.723,  Adjusted R-squared:  0.7197
## F-statistic: 217.1 on 6 and 499 DF,  p-value: < 2.2e-16
##
```

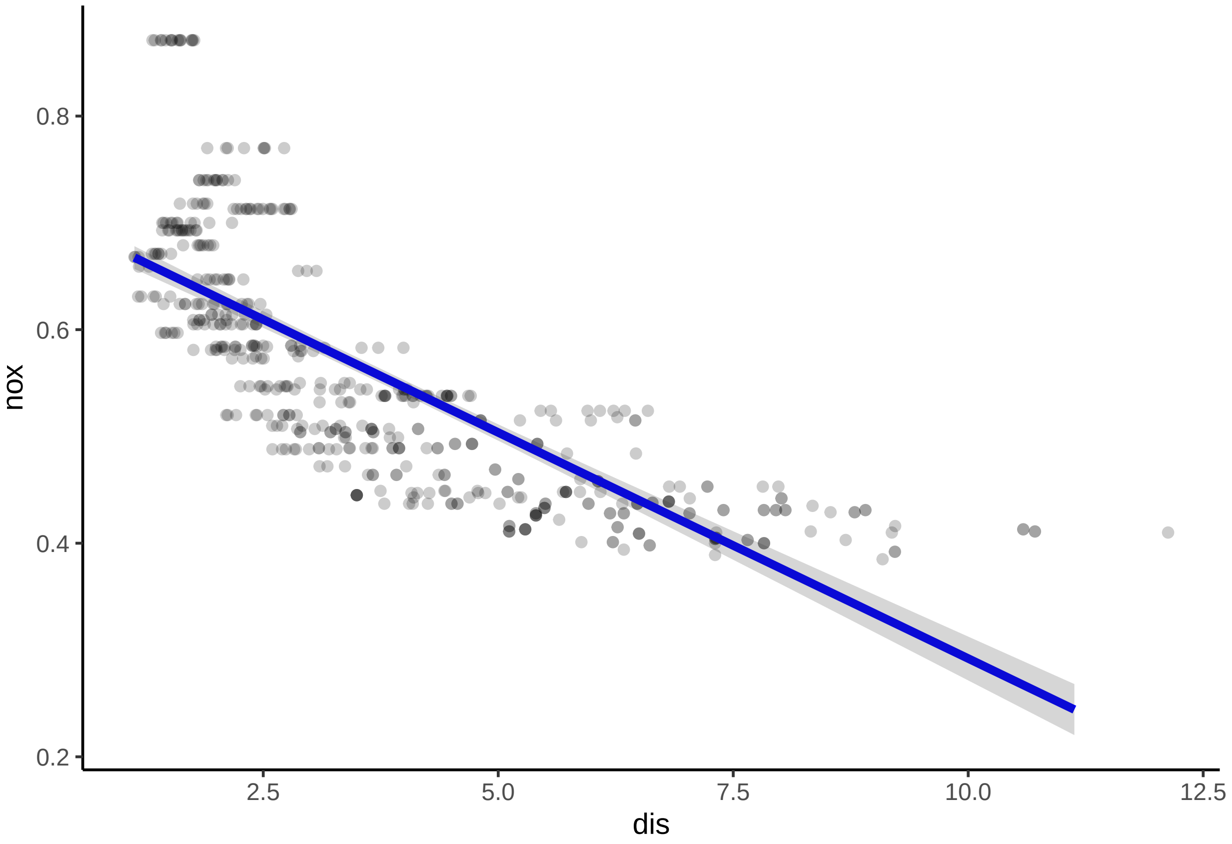
```
## Residual sum of squares 0.003809853
## ---------------------------------------------------------------
## Summary of Model fit for  7 degrees polynomial
##
## Call:
## lm(formula = nox ~ poly(dis, degrees), data = df)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.132762 -0.038635 -0.009287  0.025604  0.192399
##
## Coefficients:
##                      Estimate Std. Error t value Pr(>|t|)
## (Intercept)          0.554695   0.002709 204.748  < 2e-16 ***
## poly(dis, degrees)1 -2.003096   0.060941 -32.869  < 2e-16 ***
## poly(dis, degrees)2  0.856330   0.060941  14.052  < 2e-16 ***
## poly(dis, degrees)3 -0.318049   0.060941  -5.219 2.65e-07 ***
## poly(dis, degrees)4  0.033547   0.060941   0.550  0.58224
## poly(dis, degrees)5  0.133009   0.060941   2.183  0.02953 *
## poly(dis, degrees)6 -0.192439   0.060941  -3.158  0.00169 **
## poly(dis, degrees)7  0.169628   0.060941   2.783  0.00558 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.06094 on 498 degrees of freedom
## Multiple R-squared:  0.7273, Adjusted R-squared:  0.7234
## F-statistic: 189.7 on 7 and 498 DF,  p-value: < 2.2e-16
##
## Residual sum of squares 0.003751488
## ---------------------------------------------------------------
## Summary of Model fit for  8 degrees polynomial
##
## Call:
## lm(formula = nox ~ poly(dis, degrees), data = df)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.132328 -0.037459 -0.008615  0.023667  0.197200
##
## Coefficients:
##                      Estimate Std. Error t value Pr(>|t|)
## (Intercept)          0.554695   0.002702 205.312  < 2e-16 ***
## poly(dis, degrees)1 -2.003096   0.060774 -32.960  < 2e-16 ***
## poly(dis, degrees)2  0.856330   0.060774  14.091  < 2e-16 ***
## poly(dis, degrees)3 -0.318049   0.060774  -5.233 2.46e-07 ***
## poly(dis, degrees)4  0.033547   0.060774   0.552  0.58120
## poly(dis, degrees)5  0.133009   0.060774   2.189  0.02909 *
## poly(dis, degrees)6 -0.192439   0.060774  -3.166  0.00164 **
## poly(dis, degrees)7  0.169628   0.060774   2.791  0.00545 **
## poly(dis, degrees)8 -0.117703   0.060774  -1.937  0.05334 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.06077 on 497 degrees of freedom
```

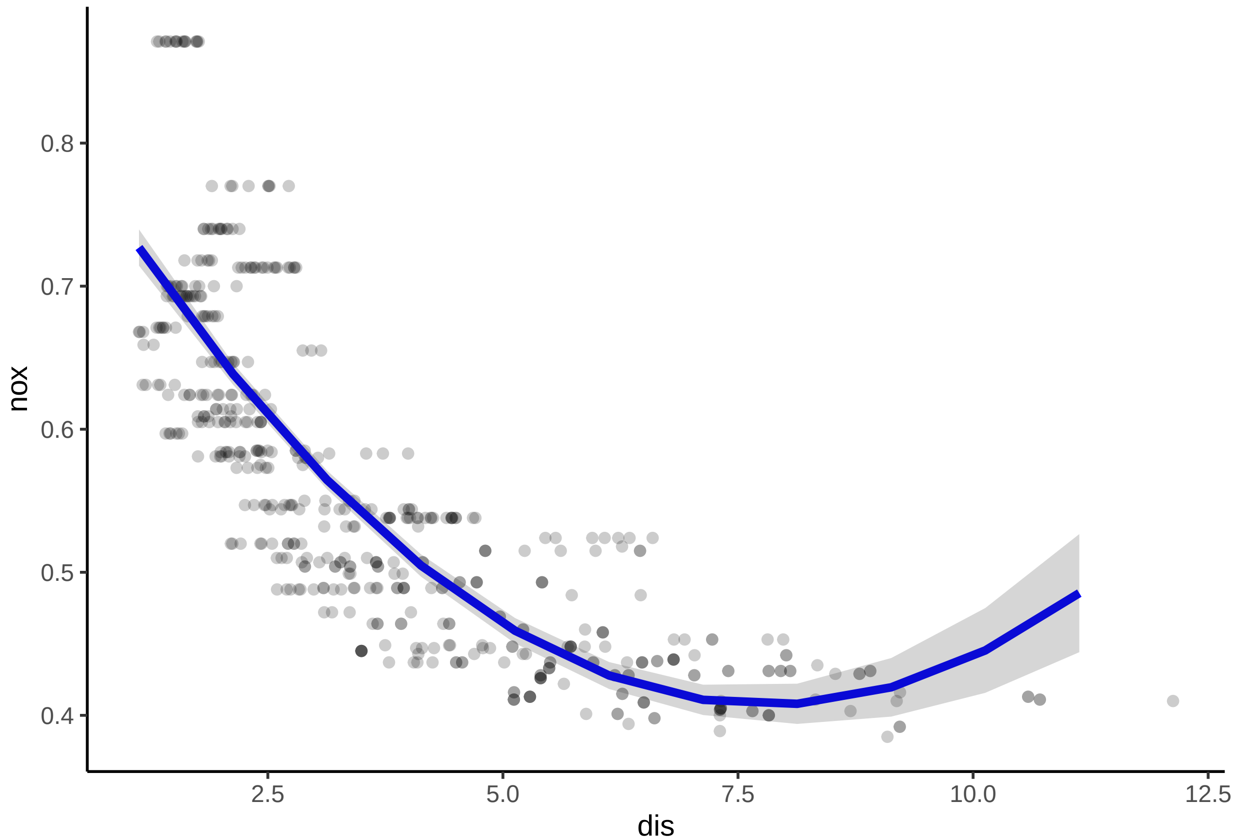
```
## Multiple R-squared:  0.7293, Adjusted R-squared:  0.7249
## F-statistic: 167.4 on 8 and 497 DF,  p-value: < 2.2e-16
##
## Residual sum of squares 0.003723387
## ------------------------------------------------------------
## Summary of Model fit for  9 degrees polynomial
##
## Call:
## lm(formula = nox ~ poly(dis, degrees), data = df)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.131023 -0.038220 -0.009488  0.023747  0.197784
##
## Coefficients:
##                      Estimate Std. Error t value Pr(>|t|)
## (Intercept)          0.554695   0.002703 205.234  < 2e-16 ***
## poly(dis, degrees)1 -2.003096   0.060797 -32.947  < 2e-16 ***
## poly(dis, degrees)2  0.856330   0.060797  14.085  < 2e-16 ***
## poly(dis, degrees)3 -0.318049   0.060797  -5.231 2.49e-07 ***
## poly(dis, degrees)4  0.033547   0.060797   0.552  0.58134
## poly(dis, degrees)5  0.133009   0.060797   2.188  0.02915 *
## poly(dis, degrees)6 -0.192439   0.060797  -3.165  0.00164 **
## poly(dis, degrees)7  0.169628   0.060797   2.790  0.00547 **
## poly(dis, degrees)8 -0.117703   0.060797  -1.936  0.05343 .
## poly(dis, degrees)9  0.047947   0.060797   0.789  0.43070
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.0608 on 496 degrees of freedom
## Multiple R-squared:  0.7296, Adjusted R-squared:  0.7247
## F-statistic: 148.7 on 9 and 496 DF,  p-value: < 2.2e-16
##
## Residual sum of squares 0.003718724
## ------------------------------------------------------------
## Summary of Model fit for  10 degrees polynomial
##
## Call:
## lm(formula = nox ~ poly(dis, degrees), data = df)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.12978 -0.03816 -0.01015  0.02420  0.19694
##
## Coefficients:
##                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)           0.554695   0.002705 205.092  < 2e-16 ***
## poly(dis, degrees)1  -2.003096   0.060839 -32.925  < 2e-16 ***
## poly(dis, degrees)2   0.856330   0.060839  14.075  < 2e-16 ***
## poly(dis, degrees)3  -0.318049   0.060839  -5.228 2.54e-07 ***
## poly(dis, degrees)4   0.033547   0.060839   0.551  0.58161
## poly(dis, degrees)5   0.133009   0.060839   2.186  0.02926 *
## poly(dis, degrees)6  -0.192439   0.060839  -3.163  0.00166 **
## poly(dis, degrees)7   0.169628   0.060839   2.788  0.00550 **
```

```
## poly(dis, degrees)8  -0.117703   0.060839  -1.935  0.05360 .
## poly(dis, degrees)9   0.047947   0.060839   0.788  0.43102
## poly(dis, degrees)10 -0.034054   0.060839  -0.560  0.57591
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.06084 on 495 degrees of freedom
## Multiple R-squared:  0.7298, Adjusted R-squared:  0.7243
## F-statistic: 133.7 on 10 and 495 DF,  p-value: < 2.2e-16
##
## Residual sum of squares 0.003716371
## ---------------------------------------------------------------
```

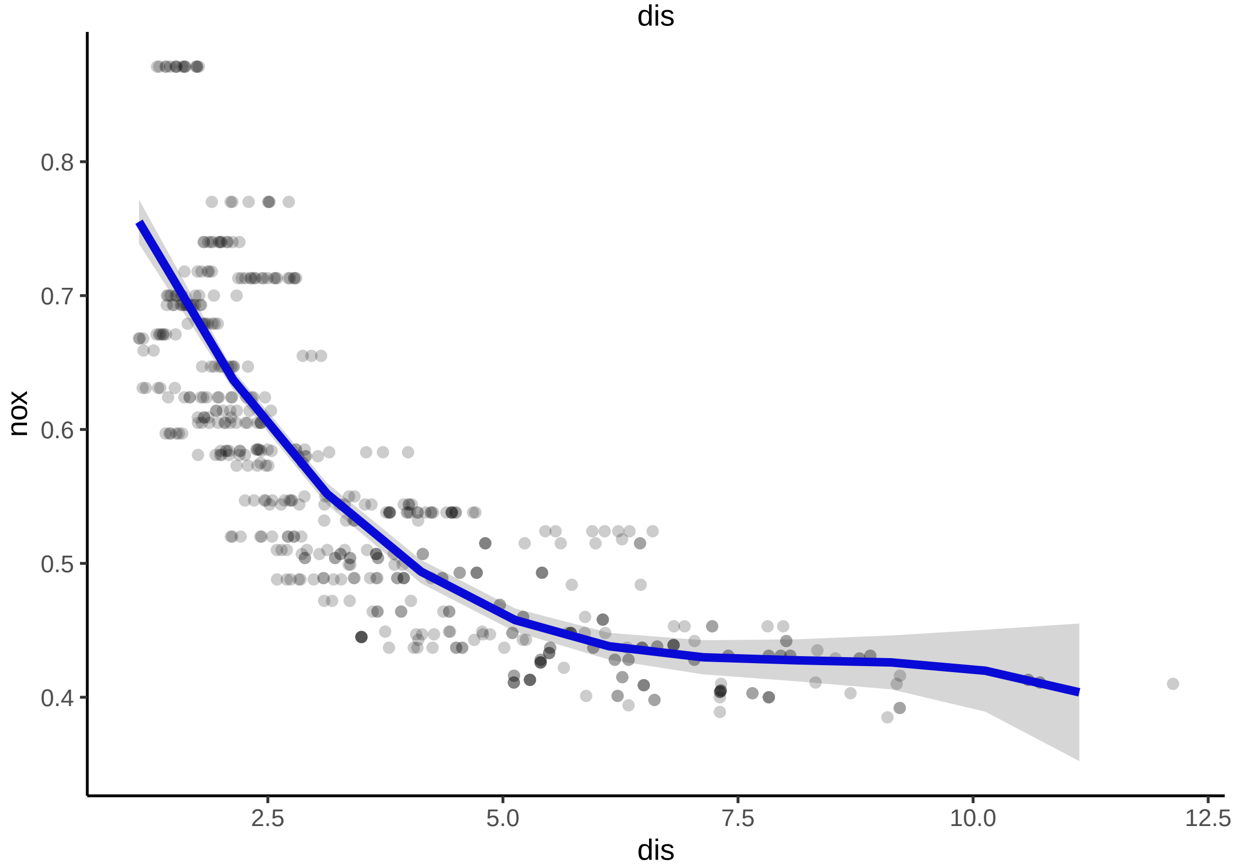
Generate plots for each poly model

```
range_poly = 1:10
for (degrees in range_poly){
  poly.fit <- lm(nox ~ poly(dis,degrees), data = df) #fit a cubic polynomial regression
  dis.range <- seq(from = min(df$dis), to = max(df$dis)) #new data made for dis
  pred <-  predict(poly.fit, newdata = list(dis = dis.range), se = TRUE) #assign predicted response to p
  conf.int <- cbind(pred$fit + 2 * pred$se.fit, pred$fit - 2 * pred$se.fit) #confidence interval
  predictions <- data.frame(DIS = dis.range, NOX = pred$fit,upper = conf.int[, 1], lower = conf.int[, 2]
  plot<- ggplot() +
        geom_point(data= df, aes(dis, nox), col = 'Black', alpha = 0.2) + #points representing observed
        geom_line(data = predictions, aes(DIS, NOX), size = 1.5, col = 'Blue') + #line representing pre
        geom_ribbon(data = predictions, aes(x = DIS, ymin = lower, ymax = upper), alpha = 0.2) + theme_c
  print(plot)
}
```

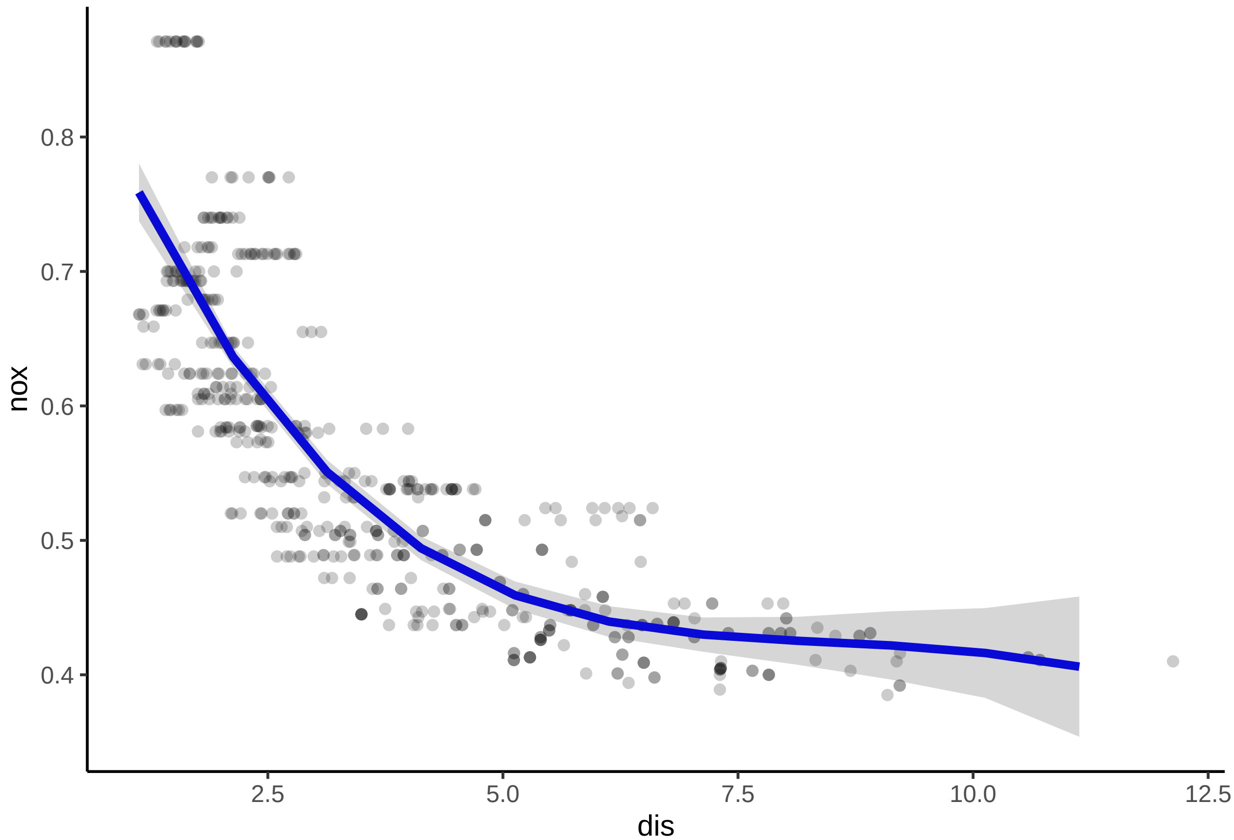


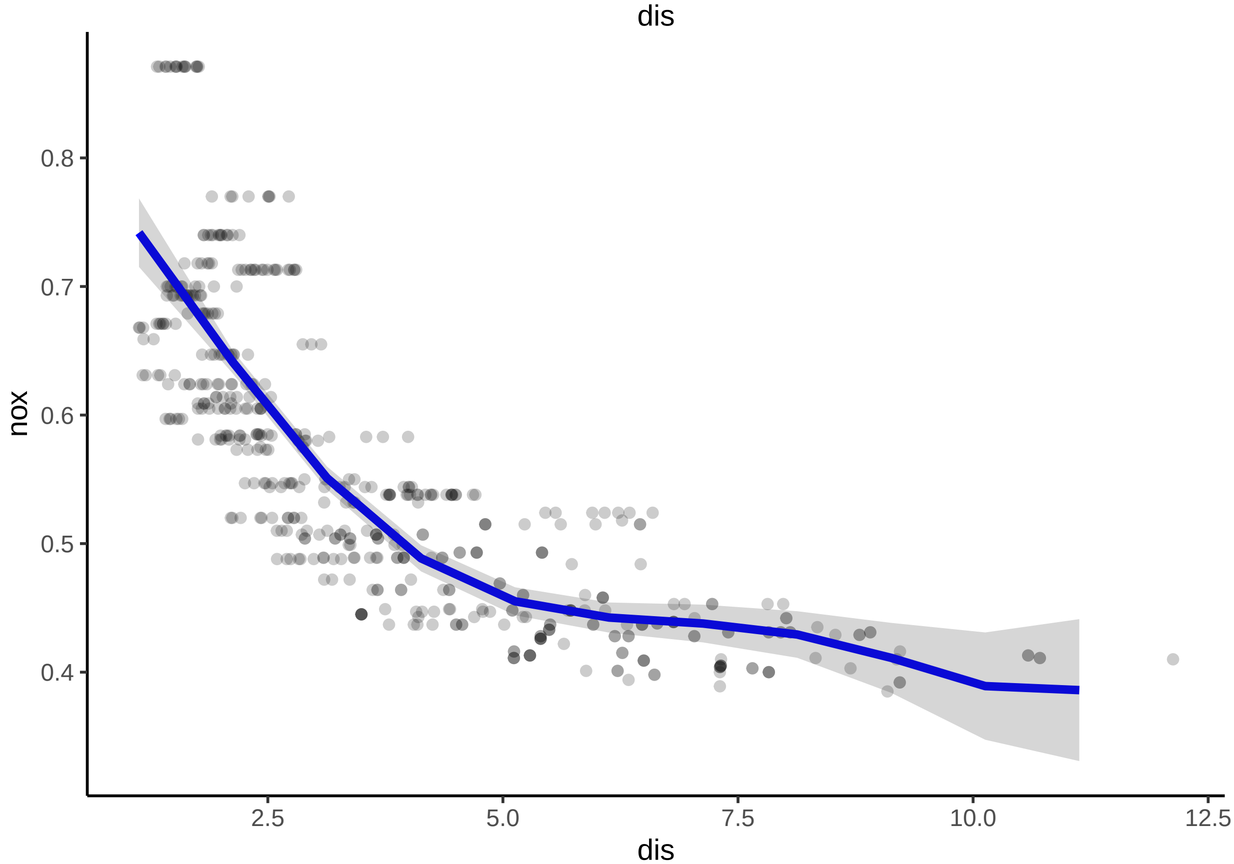


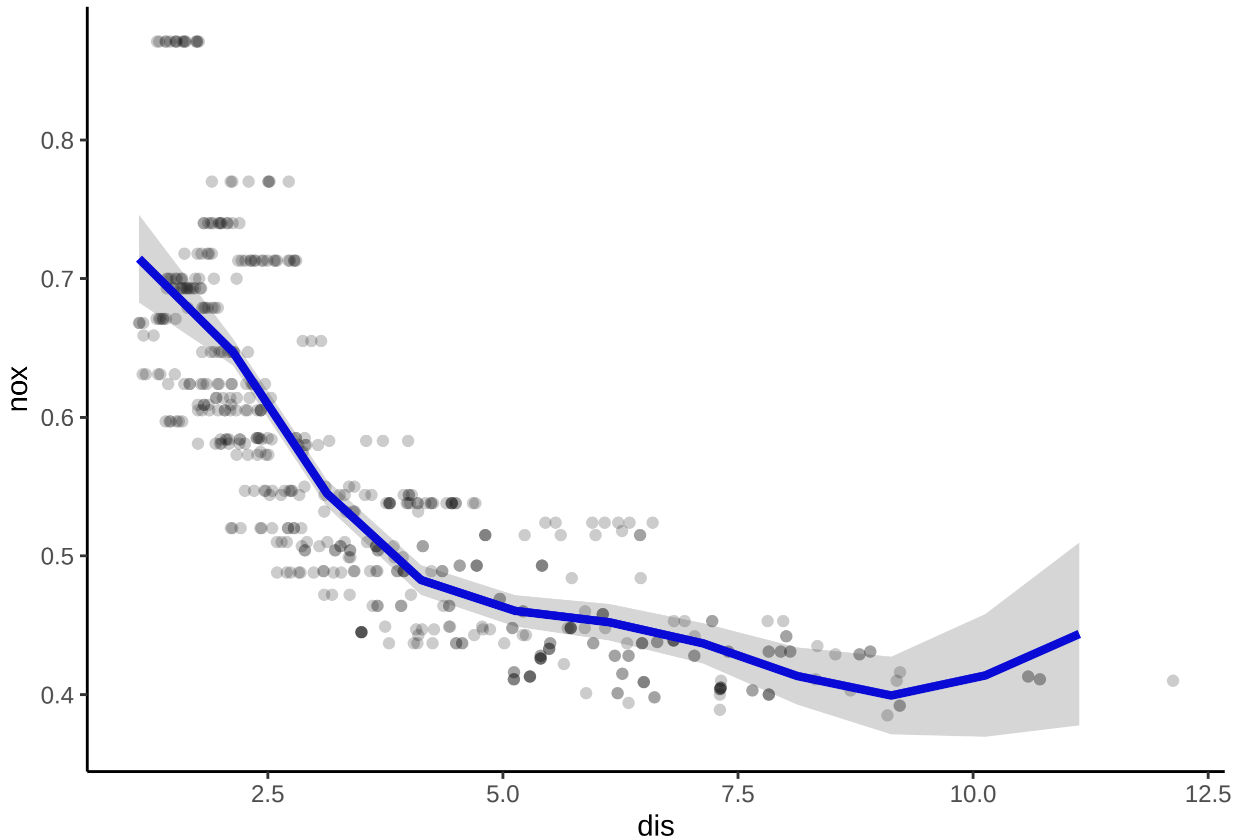
nox
0.8
0.7
0.6
0.5
0.4
2.5
5.0
7.5
10.0
12.5
dis



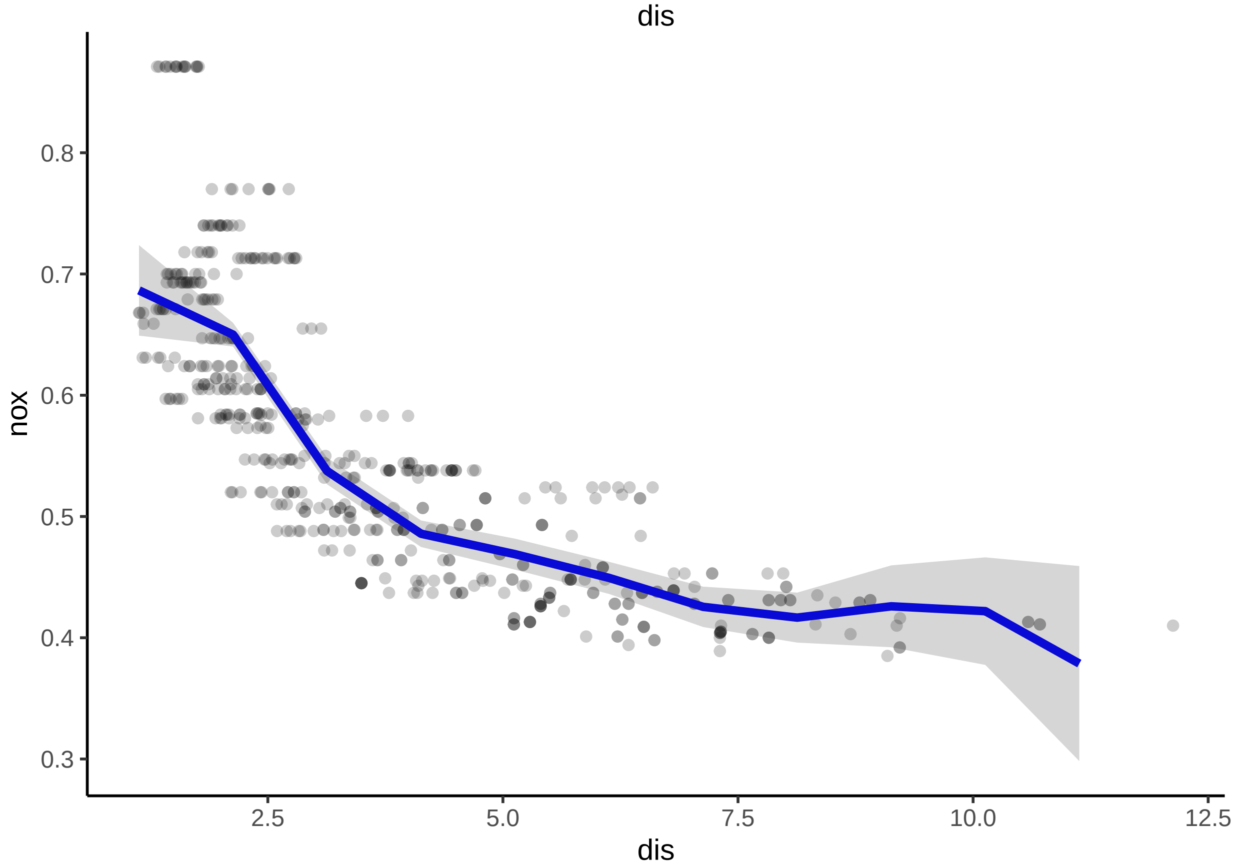
nox
0.8
0.7
0.6
0.5
0.4
2.5
5.0
7.5
10.0
12.5
dis


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

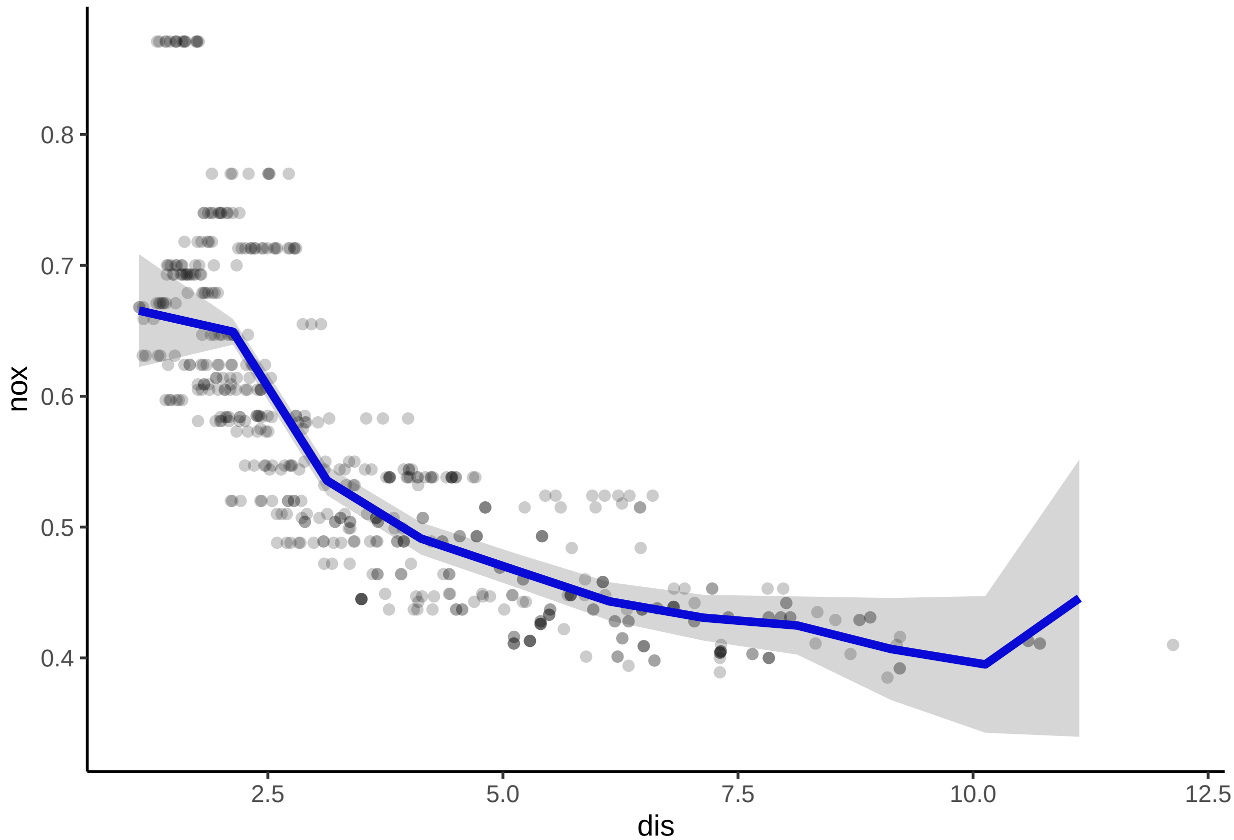

nox
0.8
0.7
0.6
0.5
0.4
2.5
5.0
7.5
10.0
12.5
dis



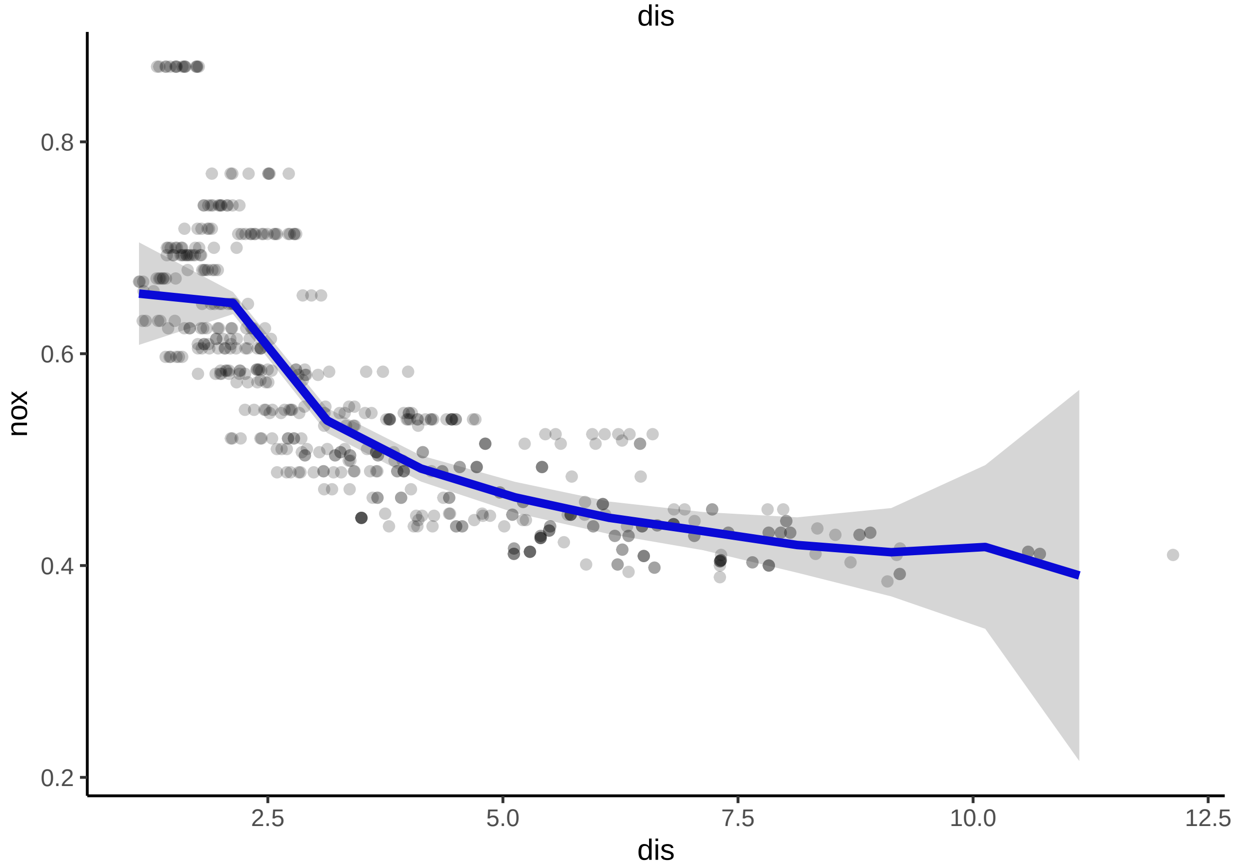
nox
0.8
0.7
0.6
0.5
0.4
2.5
5.0
7.5
10.0
12.5
dis



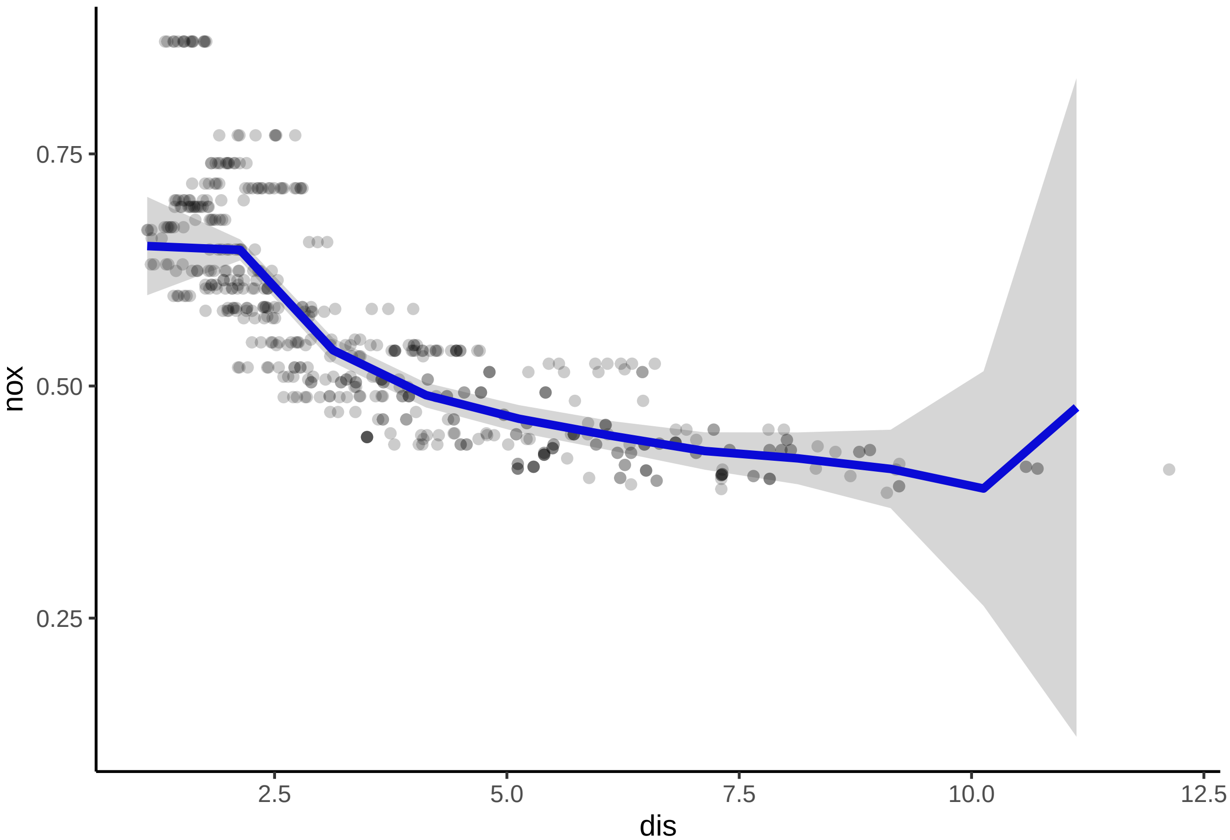
nox
0.8
0.7
0.6
0.5
0.4
0.3
2.5
5.0
7.5
10.0
12.5
dis



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



0.8
0.6
0.4
0.2
nox
2.5
5.0
7.5
10.0
12.5
dis



(c) Perform cross-validation or another approach (hint: anova) to select the optimal degree for the polynomial, and explain your results.

```
library(boot)
set.seed(123)
range_poly = 1:10

cv.error=rep(NA,10) #empty vector with a length of 10
for (degrees in range_poly){ #for degrees in 1:10
    poly.fit = glm(nox ~ poly(dis ,degrees),data=df) #fit a polynomial
    cv.error[degrees] = cv.glm(df,poly.fit, K=10)$delta[1] #delta is a vector of length two.
    #The first component is the raw cross-validation estimate of prediction error.
}
which.min(cv.error) #which degree produced the lowest MSE (delta)
```

```
## [1] 3
```

Alternatively, you can use an anova to compare the 10 degrees

```
fit.1 <- lm(nox ~ poly(dis ,1),data=df)
fit.2 <- lm(nox ~ poly(dis ,2),data=df)
fit.3 <- lm(nox ~ poly(dis ,3),data=df)
fit.4 <- lm(nox ~ poly(dis ,4),data=df)
fit.5 <- lm(nox ~ poly(dis ,5),data=df)
fit.6 <- lm(nox ~ poly(dis ,6),data=df)
fit.7 <- lm(nox ~ poly(dis ,7),data=df)
fit.8 <-lm(nox ~ poly(dis ,8),data=df)
fit.9 <- lm(nox ~ poly(dis ,9),data=df)
fit.10 <- lm(nox ~ poly(dis ,10),data=df)
```

```r
anova(fit.1,fit.2,fit.3,fit.4,fit.5,fit.6,fit.7,fit.8,fit.9,fit.10)
```

```
## Analysis of Variance Table
##
## Model  1: nox ~ poly(dis, 1)
## Model  2: nox ~ poly(dis, 2)
## Model  3: nox ~ poly(dis, 3)
## Model  4: nox ~ poly(dis, 4)
## Model  5: nox ~ poly(dis, 5)
## Model  6: nox ~ poly(dis, 6)
## Model  7: nox ~ poly(dis, 7)
## Model  8: nox ~ poly(dis, 8)
## Model  9: nox ~ poly(dis, 9)
## Model 10: nox ~ poly(dis, 10)
##    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
```

The p-values from the ANOVA indicate that we can reject the null hypothesis when compare model 1 to model 2 and model 2 to model 3. This means that both the linear and quadratic models are not sufficient to explain the data. The p-value for the comparison of model 3 and 4 is not significant so we could say that a 3rd degree polynomial is sufficient to explain the data over the 4th. In this case, 3rd degree polynomial would be sufficient to explain the data, similar to what was found using CV.

(d) Use the bs() function to fit a regression spline to predict nox using dis. Report the output for the fit using four degrees of freedom. How did you choose the knots? Plot the resulting fit.

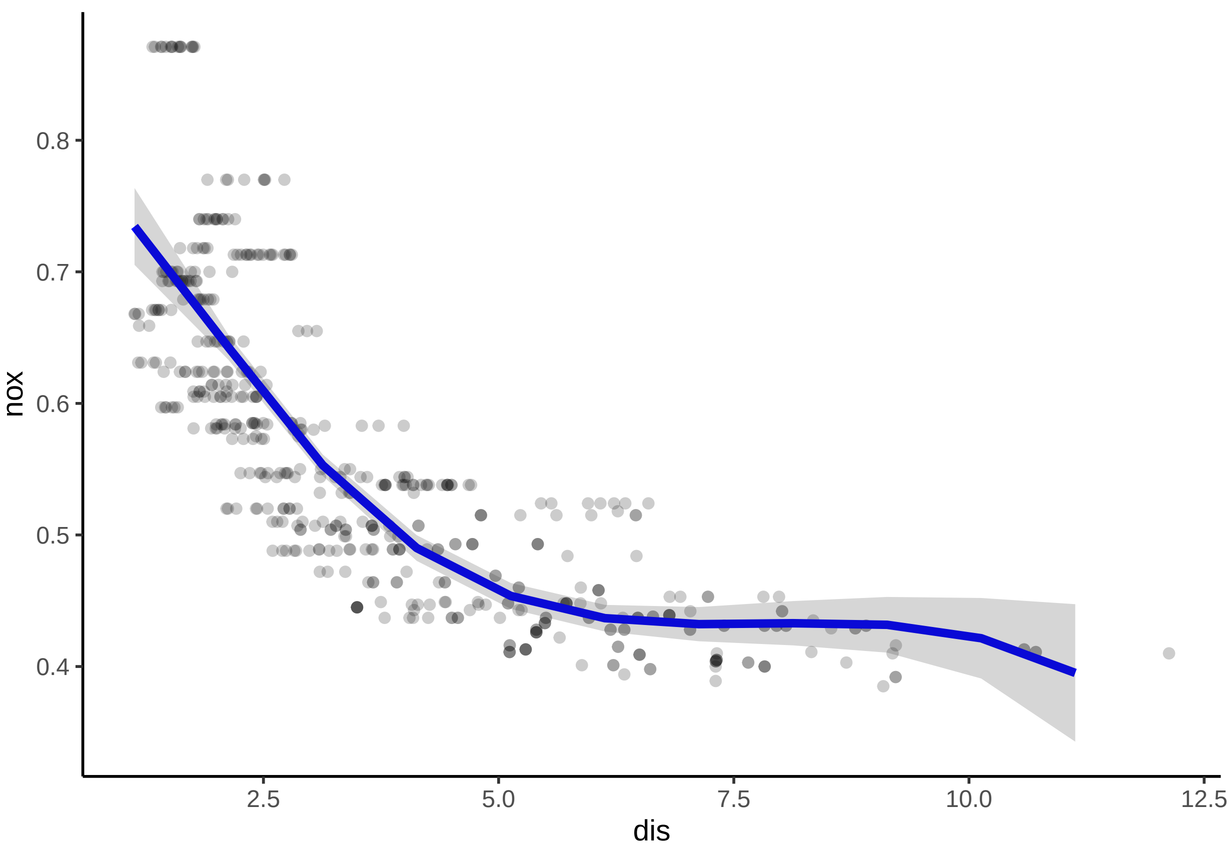
```r
library(splines) #load library for splines

spline.fit <- lm(nox ~ bs(dis,df = 4), data = df)
summary(spline.fit)
```

```
##
## Call:
## lm(formula = nox ~ bs(dis, df = 4), data = df)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.124622 -0.039259 -0.008514  0.020850  0.193891
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)       0.73447    0.01460  50.306  < 2e-16 ***
```

```
## bs(dis, df = 4)1 -0.05810    0.02186  -2.658  0.00812 **
## bs(dis, df = 4)2 -0.46356    0.02366 -19.596  < 2e-16 ***
## bs(dis, df = 4)3 -0.19979    0.04311  -4.634 4.58e-06 ***
## bs(dis, df = 4)4 -0.38881    0.04551  -8.544  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.06195 on 501 degrees of freedom
## Multiple R-squared:  0.7164, Adjusted R-squared:  0.7142
## F-statistic: 316.5 on 4 and 501 DF,  p-value: < 2.2e-16
```

If we dont have a specific location we'd like to put our knots, we could use the df option to produce a spline with knots at uniform quantiles of the data. The model finds each knot(1,2,3 and 4) of the dis coefficient to be statistically significant (p<0.05).

```
dis.range <- seq(from = min(df$dis), to = max(df$dis))
pred <-  predict(spline.fit, newdata = list(dis = dis.range), se = TRUE) #assign predicted response to p
conf.int <- cbind(pred$fit + 2 * pred$se.fit, pred$fit - 2 * pred$se.fit)
predictions <- data.frame(DIS = dis.range, NOX = pred$fit,upper = conf.int[, 1], lower = conf.int[, 2])
#install.packages('ggplot2') install ggplot2 if you havent already done so
library(ggplot2)
ggplot() +
    geom_point(data= df, aes(dis, nox), col = 'Black', alpha = 0.2) + #points representing observed val
    geom_line(data = predictions, aes(DIS, NOX), size = 1.5, col = 'Blue') + #line representing predict
    geom_ribbon(data = predictions, aes(x = DIS, ymin = lower, ymax = upper), alpha = 0.2) + theme_class
```



(e) Now fit a regression spline for a range of degrees of freedom, and plot the resulting fits and report the resulting RSS. Describe the results obtained.

```r
range_splines = 3:10
for (spline in range_splines){
  spline.fit <- lm(nox ~ bs(dis,df = spline), data = df) #fit a spline model
  dis.range <- seq(from = min(df$dis), to = max(df$dis)) #new data made for dis
  pred <-  predict(spline.fit, newdata = list(dis = dis.range), se = TRUE) #assign predicted response t
  conf.int <- cbind(pred$fit + 2 * pred$se.fit, pred$fit - 2 * pred$se.fit) #confidence interval
  predictions <- data.frame(DIS = dis.range, NOX = pred$fit,upper = conf.int[, 1], lower = conf.int[, 2
  cat("Summary of Model fit for ",spline, "degrees of freedom spline regression", "\n") #give each outp
  print(summary(spline.fit)) # print summary of output
  errors <- sum(spline.fit$residuals ^2) / (nrow(df) - ncol(df))
  cat("Residual sum of squares", errors, "\n")
  cat("------------------------------------------------------------- \n") #print a line to separate ou
}
```

```
## Summary of Model fit for  3 degrees of freedom spline regression
##
## Call:
## lm(formula = nox ~ bs(dis, df = spline), data = df)
##
## 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)           0.755153   0.008283  91.168  < 2e-16 ***
## bs(dis, df = spline)1 -0.498271   0.032542 -15.312  < 2e-16 ***
## bs(dis, df = spline)2 -0.233520   0.036994  -6.312 6.05e-10 ***
## bs(dis, df = spline)3 -0.382680   0.045455  -8.419 4.00e-16 ***
## ---
## 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
##
## Residual sum of squares 0.003923137
## -------------------------------------------------------------
## Summary of Model fit for  4 degrees of freedom spline regression
##
## Call:
## lm(formula = nox ~ bs(dis, df = spline), data = df)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.124622 -0.039259 -0.008514  0.020850  0.193891
##
## Coefficients:
##                      Estimate Std. Error t value Pr(>|t|)
## (Intercept)           0.73447    0.01460  50.306  < 2e-16 ***
## bs(dis, df = spline)1 -0.05810    0.02186  -2.658  0.00812 **
## bs(dis, df = spline)2 -0.46356    0.02366 -19.596  < 2e-16 ***
## bs(dis, df = spline)3 -0.19979    0.04311  -4.634 4.58e-06 ***
## bs(dis, df = spline)4 -0.38881    0.04551  -8.544  < 2e-16 ***
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.06195 on 501 degrees of freedom
## Multiple R-squared:  0.7164, Adjusted R-squared:  0.7142
## F-statistic: 316.5 on 4 and 501 DF,  p-value: < 2.2e-16
##
## Residual sum of squares 0.003900152
## ---------------------------------------------------------------
## Summary of Model fit for  5 degrees of freedom spline regression
##
## Call:
## lm(formula = nox ~ bs(dis, df = spline), data = df)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.132814 -0.039097 -0.008521  0.023493  0.197761
##
## Coefficients:
##                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)            0.67248    0.01996  33.699  < 2e-16 ***
## bs(dis, df = spline)1  0.08311    0.02875   2.891  0.00401 **
## bs(dis, df = spline)2 -0.13460    0.01985  -6.782 3.35e-11 ***
## bs(dis, df = spline)3 -0.25505    0.03477  -7.336 8.95e-13 ***
## bs(dis, df = spline)4 -0.26785    0.04059  -6.599 1.06e-10 ***
## bs(dis, df = spline)5 -0.26103    0.05214  -5.007 7.69e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.06067 on 500 degrees of freedom
## Multiple R-squared:  0.7286, Adjusted R-squared:  0.7259
## F-statistic: 268.5 on 5 and 500 DF,  p-value: < 2.2e-16
##
## Residual sum of squares 0.003732602
## ---------------------------------------------------------------
## Summary of Model fit for  6 degrees of freedom spline regression
##
## Call:
## lm(formula = nox ~ bs(dis, df = spline), data = df)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.128538 -0.037813 -0.009987  0.022644  0.195494
##
## Coefficients:
##                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)            0.65622    0.02370  27.689  < 2e-16 ***
## bs(dis, df = spline)1  0.10222    0.03516   2.907  0.00381 **
## bs(dis, df = spline)2 -0.02963    0.02338  -1.267  0.20571
## bs(dis, df = spline)3 -0.15959    0.02791  -5.718 1.86e-08 ***
## bs(dis, df = spline)4 -0.22815    0.03324  -6.864 1.99e-11 ***
## bs(dis, df = spline)5 -0.26272    0.04930  -5.329 1.50e-07 ***
## bs(dis, df = spline)6 -0.24002    0.05434  -4.417 1.23e-05 ***
## ---
```

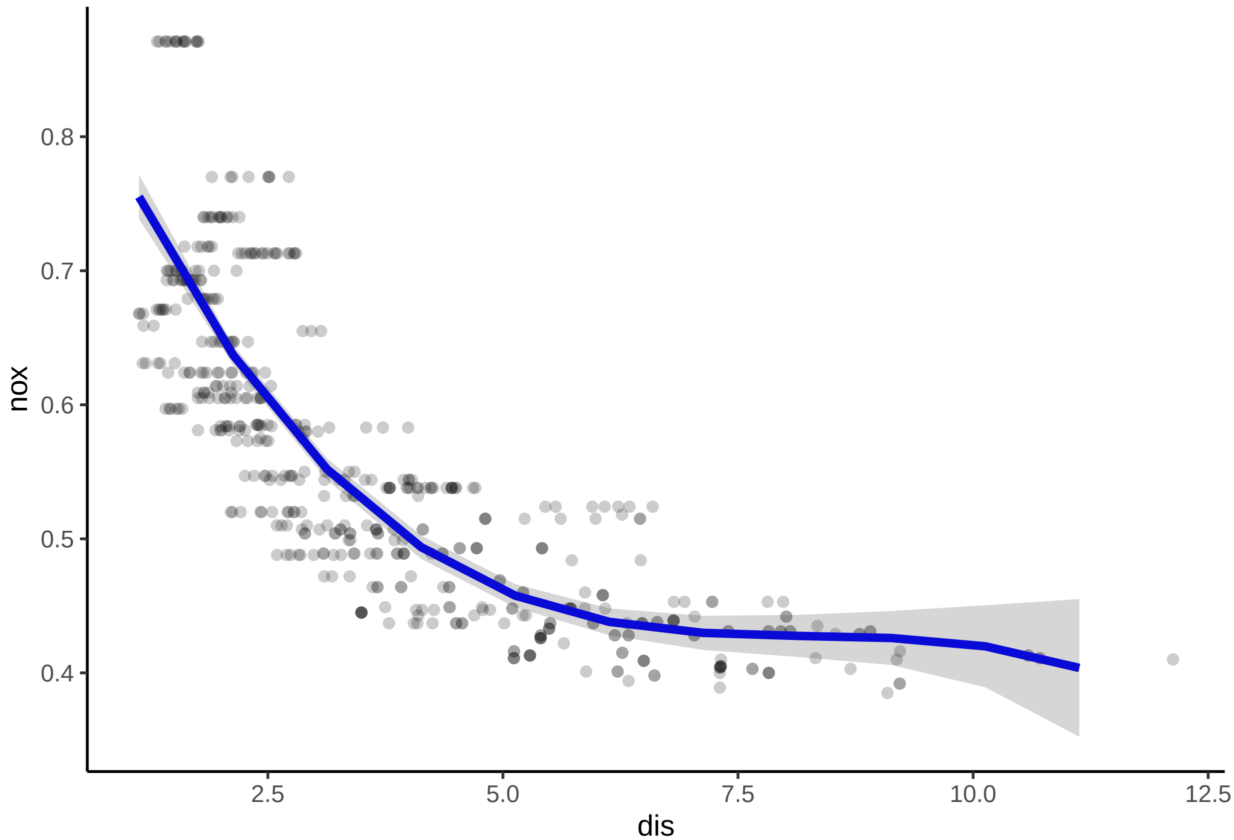
```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.06062 on 499 degrees of freedom
## Multiple R-squared:  0.7295, Adjusted R-squared:  0.7263
## F-statistic: 224.3 on 6 and 499 DF,  p-value: < 2.2e-16
##
## Residual sum of squares 0.003720012
## --------------------------------------------------------------
## Summary of Model fit for  7 degrees of freedom spline regression
##
## Call:
## lm(formula = nox ~ bs(dis, df = spline), data = df)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.12702 -0.03821 -0.01068  0.02296  0.19579
##
## Coefficients:
##                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)            0.64558    0.02633  24.516  < 2e-16 ***
## bs(dis, df = spline)1  0.11238    0.04098   2.742  0.00632 **
## bs(dis, df = spline)2  0.02461    0.02638   0.933  0.35138
## bs(dis, df = spline)3 -0.09216    0.03119  -2.955  0.00327 **
## bs(dis, df = spline)4 -0.16212    0.02829  -5.731 1.73e-08 ***
## bs(dis, df = spline)5 -0.22224    0.03873  -5.738 1.66e-08 ***
## bs(dis, df = spline)6 -0.24885    0.05147  -4.834 1.78e-06 ***
## bs(dis, df = spline)7 -0.23091    0.05779  -3.995 7.44e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.06062 on 498 degrees of freedom
## Multiple R-squared:  0.7301, Adjusted R-squared:  0.7264
## F-statistic: 192.5 on 7 and 498 DF,  p-value: < 2.2e-16
##
## Residual sum of squares 0.003711733
## --------------------------------------------------------------
## Summary of Model fit for  8 degrees of freedom spline regression
##
## Call:
## lm(formula = nox ~ bs(dis, df = spline), data = df)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.12576 -0.03773 -0.01012  0.02562  0.18982
##
## Coefficients:
##                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)            0.63234    0.02793  22.642  < 2e-16 ***
## bs(dis, df = spline)1  0.13966    0.04450   3.139 0.001799 **
## bs(dis, df = spline)2  0.03656    0.02870   1.274 0.203308
## bs(dis, df = spline)3 -0.01656    0.03280  -0.505 0.613734
## bs(dis, df = spline)4 -0.13408    0.03049  -4.398 1.34e-05 ***
## bs(dis, df = spline)5 -0.14378    0.03177  -4.525 7.55e-06 ***
## bs(dis, df = spline)6 -0.23669    0.04022  -5.885 7.31e-09 ***
```

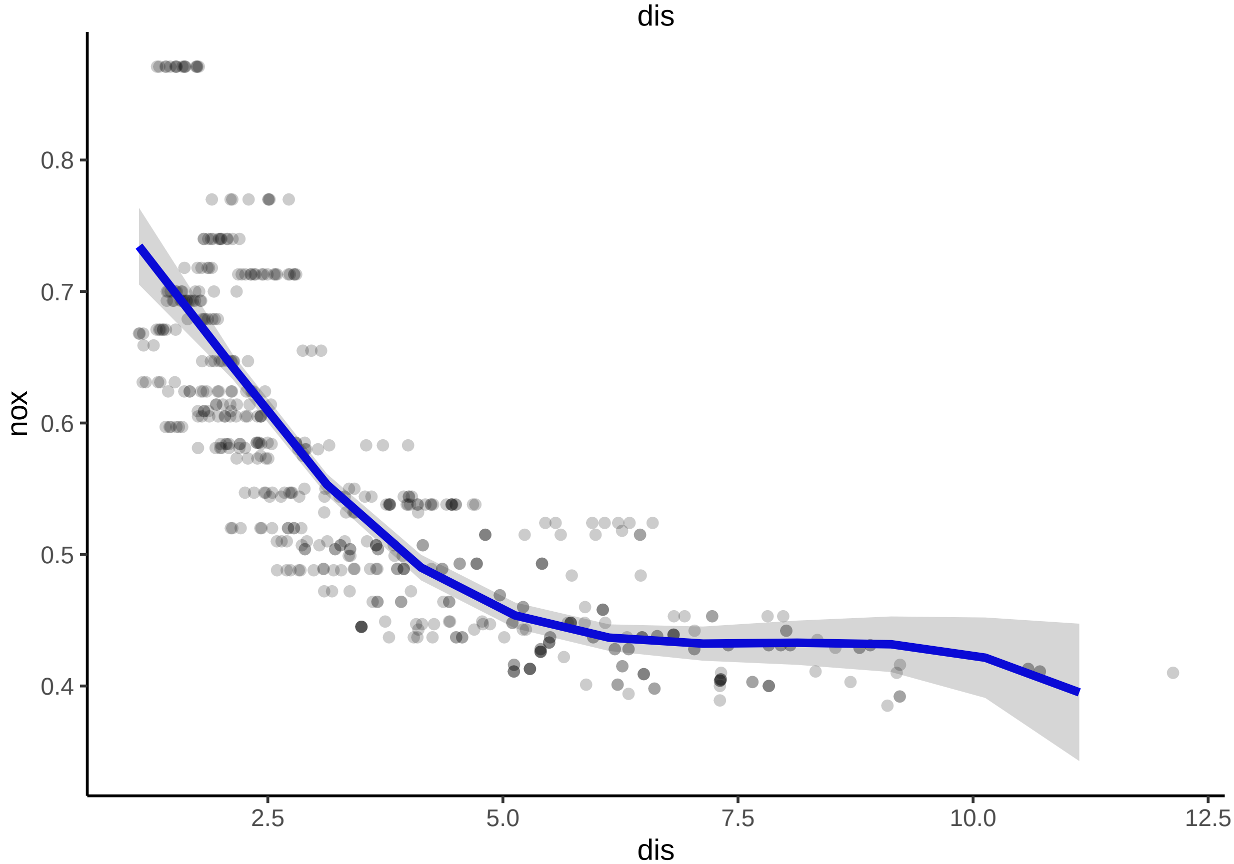
```
## bs(dis, df = spline)7 -0.20770    0.05654  -3.674 0.000265 ***
## bs(dis, df = spline)8 -0.22869    0.05938  -3.851 0.000133 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.06046 on 497 degrees of freedom
## Multiple R-squared:  0.732,  Adjusted R-squared:  0.7277
## F-statistic: 169.7 on 8 and 497 DF,  p-value: < 2.2e-16
##
## Residual sum of squares 0.003685588
## ----------------------------------------------------------------
## Summary of Model fit for  9 degrees of freedom spline regression
##
## Call:
## lm(formula = nox ~ bs(dis, df = spline), data = df)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.12946 -0.03950 -0.01056  0.02824  0.19559
##
## Coefficients:
##                        Estimate Std. Error t value Pr(>|t|)
## (Intercept)            0.633195   0.029181  21.699  < 2e-16 ***
## bs(dis, df = spline)1  0.130442   0.048427   2.694 0.007308 **
## bs(dis, df = spline)2  0.053414   0.030967   1.725 0.085174 .
## bs(dis, df = spline)3  0.004425   0.034405   0.129 0.897703
## bs(dis, df = spline)4 -0.087034   0.032245  -2.699 0.007188 **
## bs(dis, df = spline)5 -0.133402   0.033137  -4.026 6.57e-05 ***
## bs(dis, df = spline)6 -0.164008   0.032510  -5.045 6.38e-07 ***
## bs(dis, df = spline)7 -0.221244   0.043497  -5.086 5.19e-07 ***
## bs(dis, df = spline)8 -0.227141   0.059743  -3.802 0.000161 ***
## bs(dis, df = spline)9 -0.221607   0.061374  -3.611 0.000336 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.06067 on 496 degrees of freedom
## Multiple R-squared:  0.7308, Adjusted R-squared:  0.7259
## F-statistic: 149.6 on 9 and 496 DF,  p-value: < 2.2e-16
##
## Residual sum of squares 0.003703149
## ----------------------------------------------------------------
## Summary of Model fit for  10 degrees of freedom spline regression
##
## Call:
## lm(formula = nox ~ bs(dis, df = spline), data = df)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.12048 -0.03810 -0.01054  0.02579  0.18254
##
## Coefficients:
##                         Estimate Std. Error t value Pr(>|t|)
## (Intercept)             0.64559    0.02975  21.697  < 2e-16 ***
## bs(dis, df = spline)1   0.07833    0.05112   1.532 0.126138
```

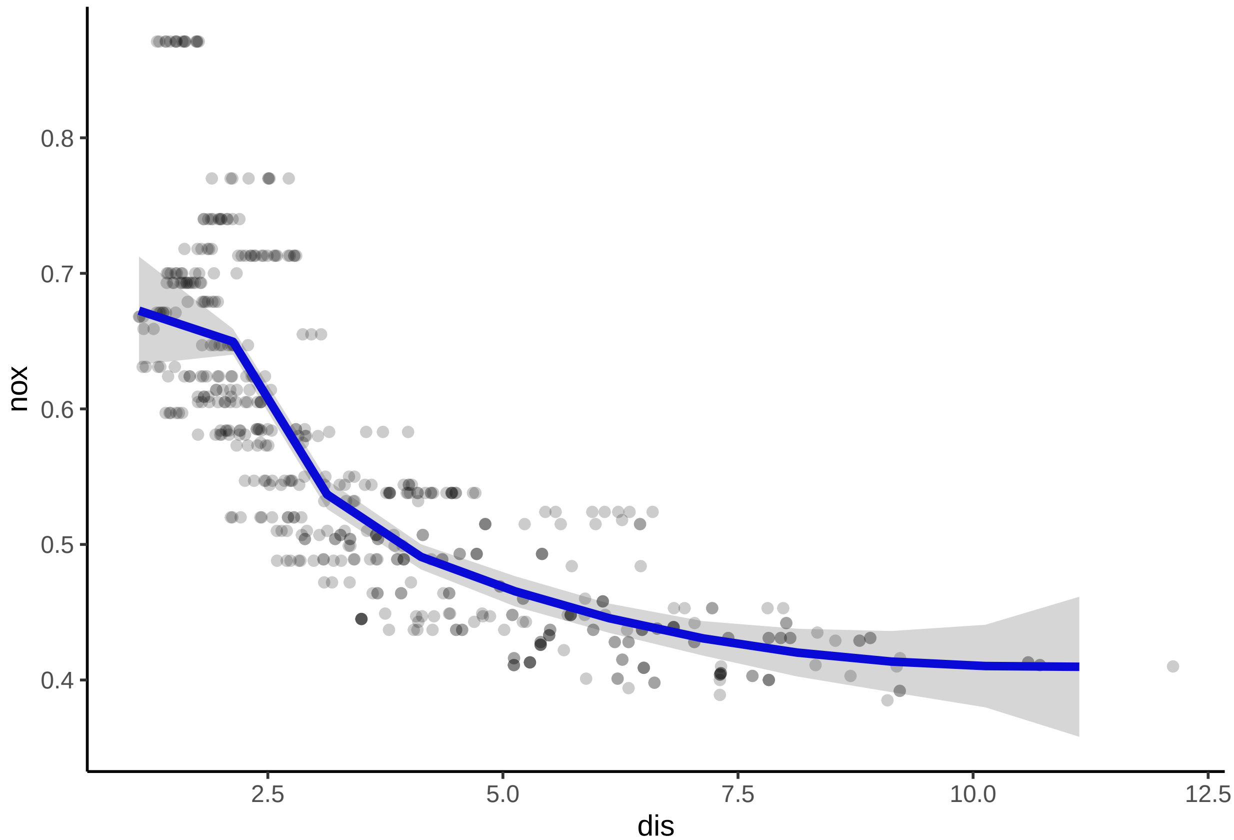
```
## bs(dis, df = spline)2   0.09019    0.03270   2.758 0.006032 **
## bs(dis, df = spline)3  -0.02698    0.03523  -0.766 0.444114
## bs(dis, df = spline)4  -0.01916    0.03279  -0.584 0.559342
## bs(dis, df = spline)5  -0.16758    0.03487  -4.806 2.05e-06 ***
## bs(dis, df = spline)6  -0.12349    0.03350  -3.686 0.000253 ***
## bs(dis, df = spline)7  -0.20389    0.03348  -6.089 2.28e-09 ***
## bs(dis, df = spline)8  -0.19998    0.04400  -4.545 6.90e-06 ***
## bs(dis, df = spline)9  -0.27818    0.06334  -4.392 1.38e-05 ***
## bs(dis, df = spline)10 -0.21977    0.06190  -3.550 0.000421 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.06018 on 495 degrees of freedom
## Multiple R-squared:  0.7357, Adjusted R-squared:  0.7303
## F-statistic: 137.8 on 10 and 495 DF,  p-value: < 2.2e-16
##
## Residual sum of squares 0.003635973
## -----------------------------------------------------------------
```

Generate plots for each spline model

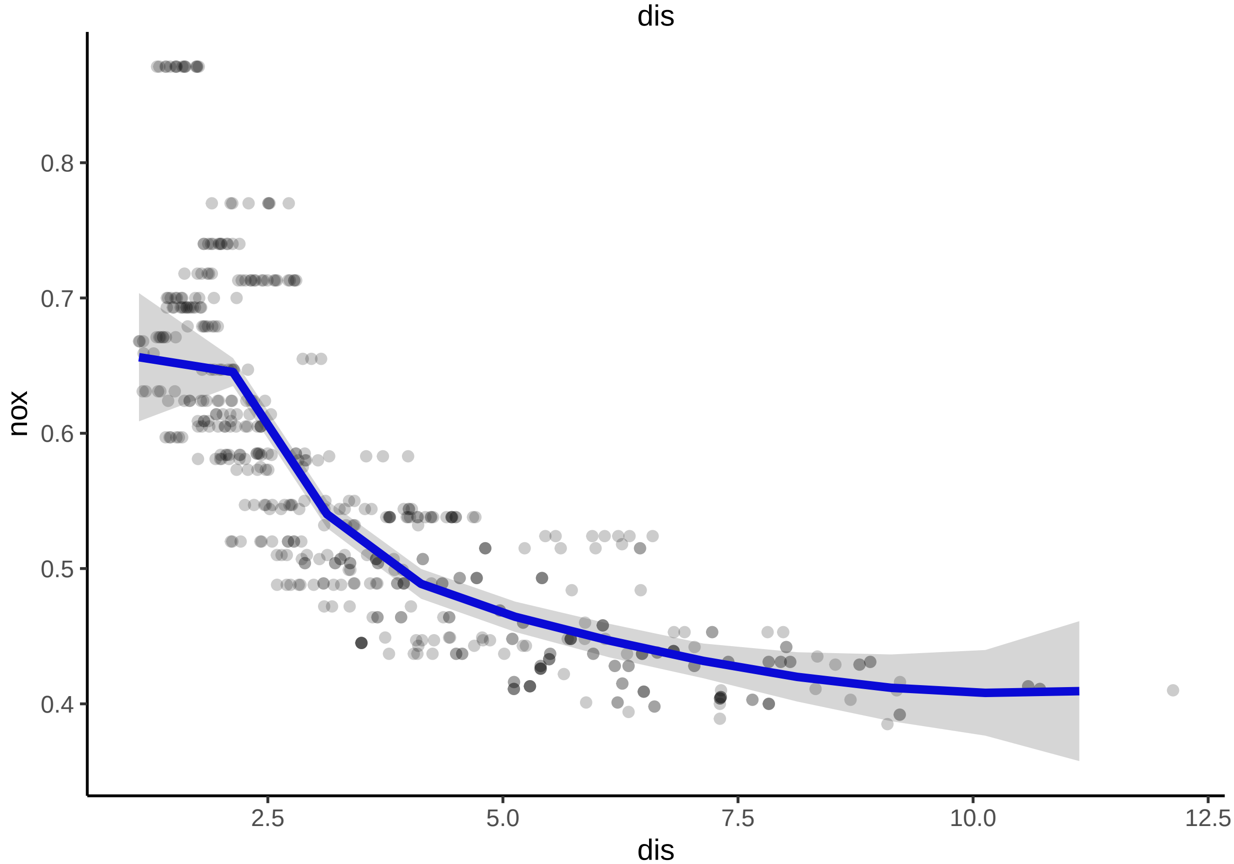
```
range_splines = 3:10
for (spline in range_splines){
  spline.fit <- lm(nox ~ bs(dis,df = spline), data = df) #fit a spline model
  dis.range <- seq(from = min(df$dis), to = max(df$dis)) #new data made for dis
  pred <-  predict(spline.fit, newdata = list(dis = dis.range), se = TRUE) #assign predicted response t
  conf.int <- cbind(pred$fit + 2 * pred$se.fit, pred$fit - 2 * pred$se.fit) #confidence interval
  predictions <- data.frame(DIS = dis.range, NOX = pred$fit,upper = conf.int[, 1], lower = conf.int[, 2
  plot<- ggplot() +
        geom_point(data= df, aes(dis, nox), col = 'Black', alpha = 0.2) + #points representing observed
        geom_line(data = predictions, aes(DIS, NOX), size = 1.5, col = 'Blue') + #line representing pre
        geom_ribbon(data = predictions, aes(x = DIS, ymin = lower, ymax = upper), alpha = 0.2) + theme_
  print(plot)
}
```


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

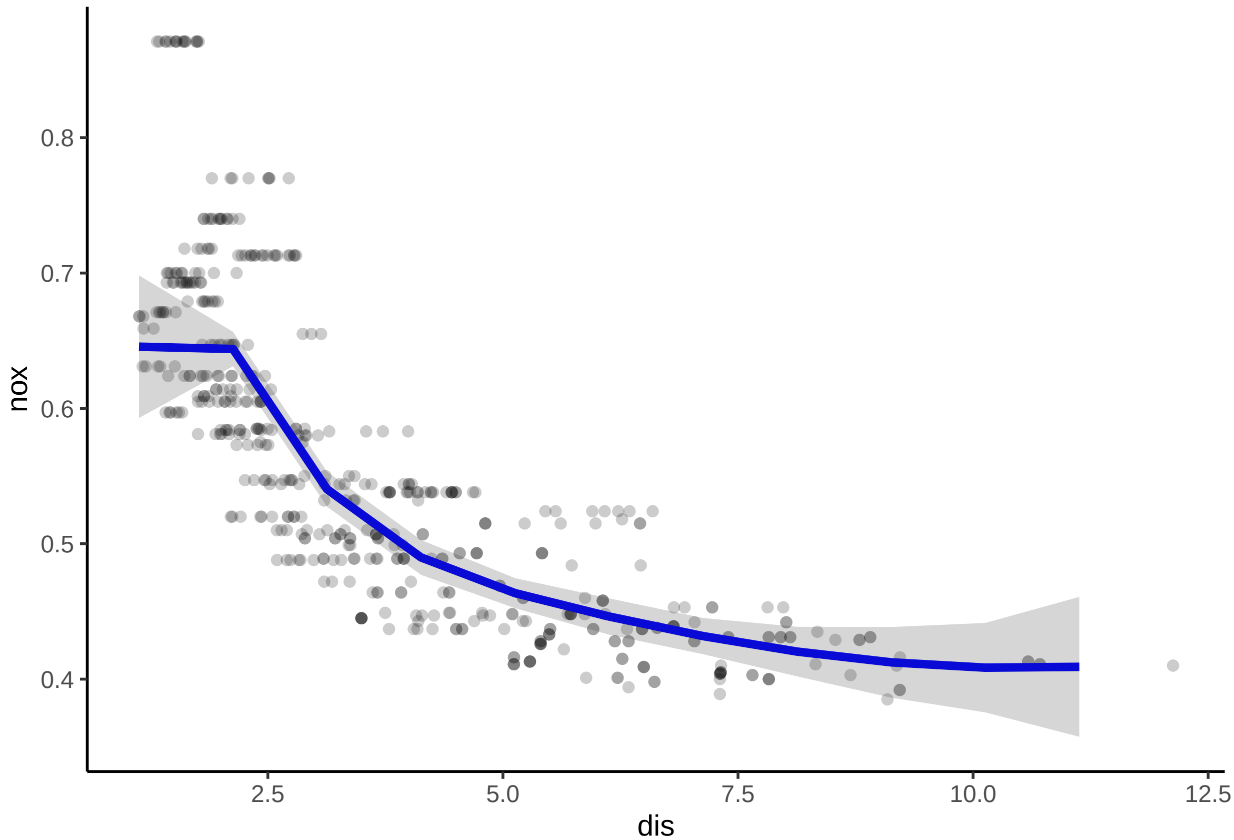

0.8
0.7
0.6
0.5
0.4
nox
2.5
5.0
7.5
10.0
12.5
dis



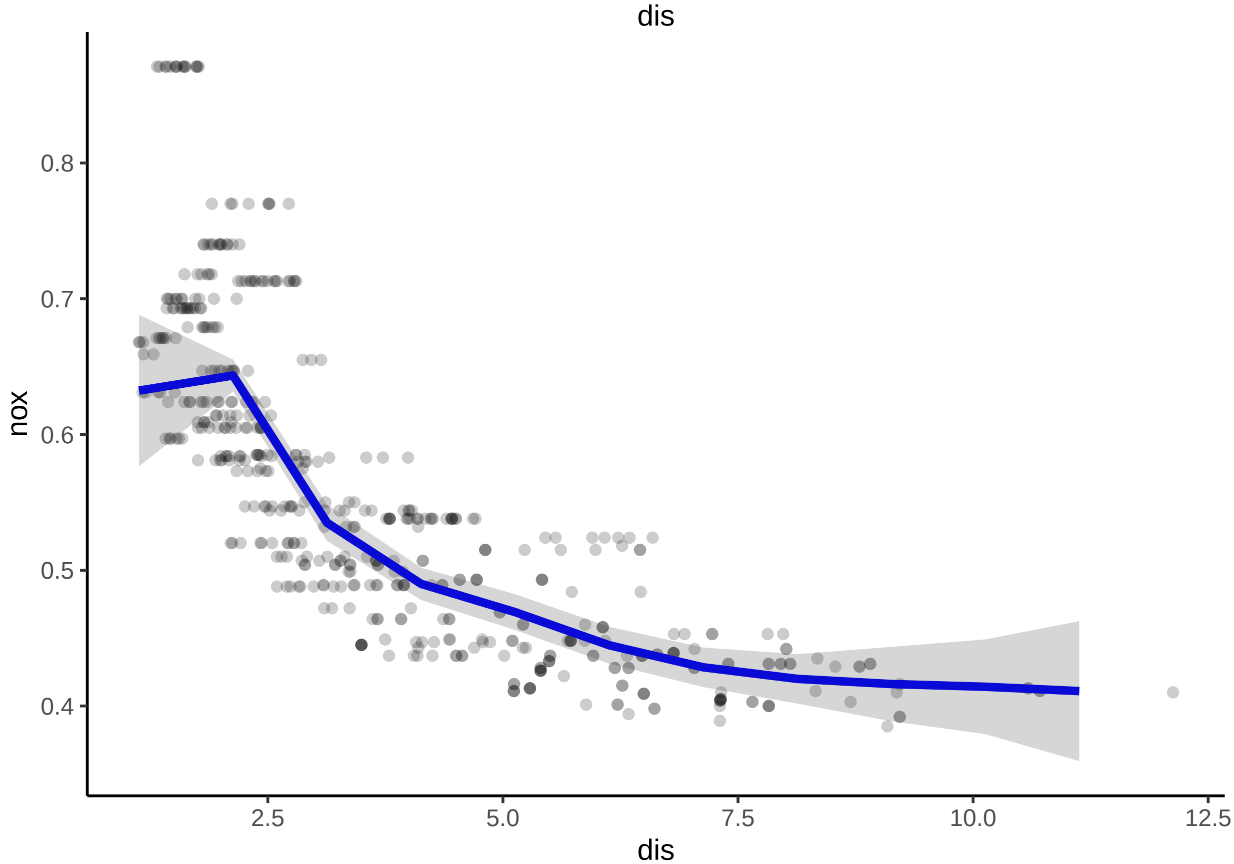
0.8
0.7
0.6
0.5
0.4
nox
2.5
5.0
7.5
10.0
12.5
dis



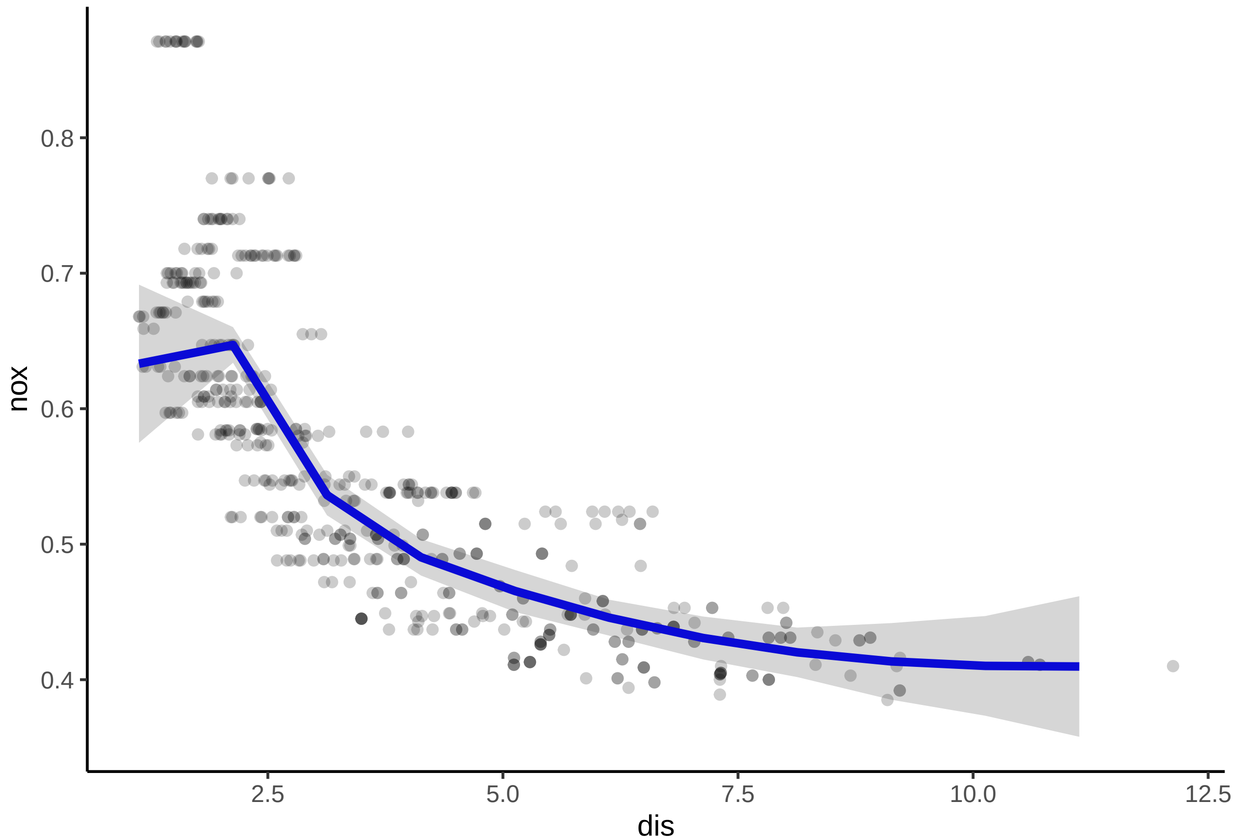
0.8
0.7
0.6
0.5
0.4
nox
2.5
5.0
7.5
10.0
12.5
dis



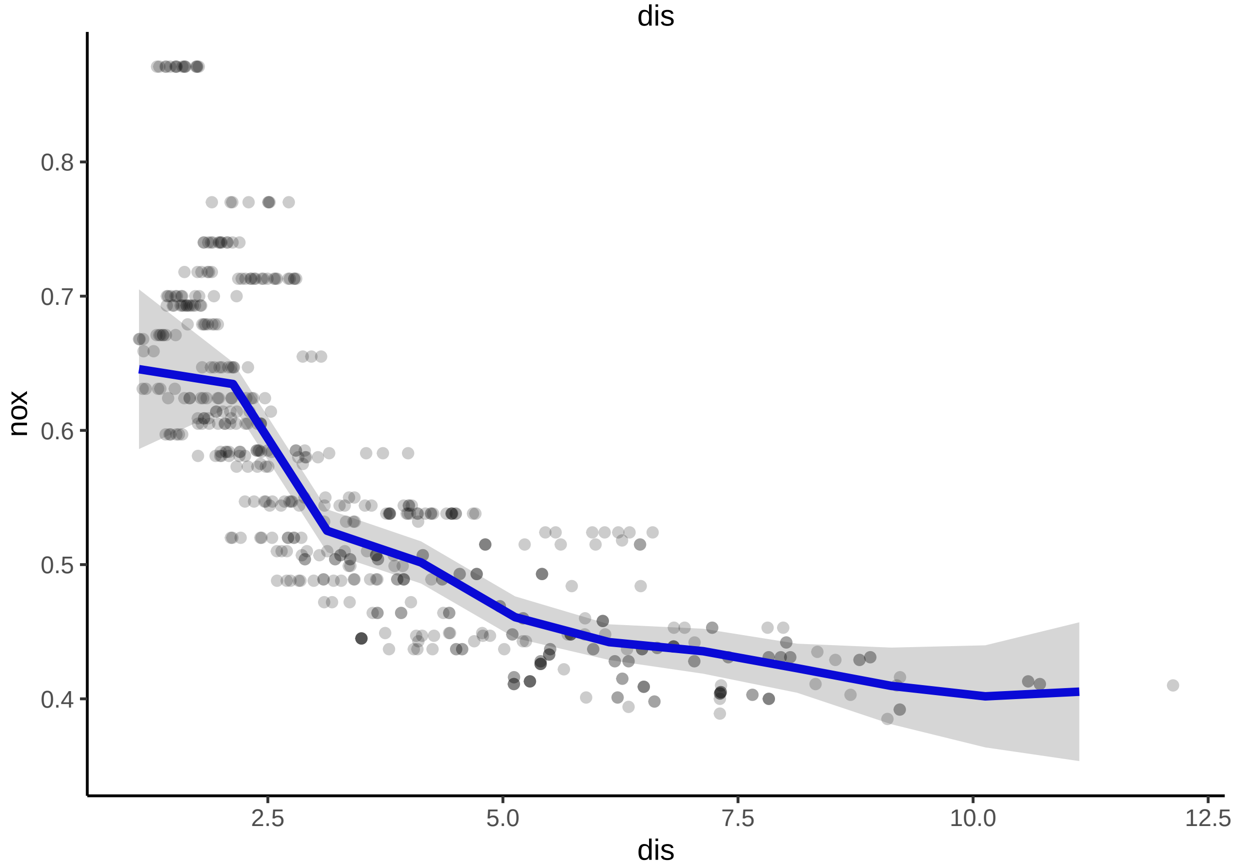
0.8
0.7
0.6
0.5
0.4
nox
2.5
5.0
7.5
10.0
12.5
dis



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



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



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

(f) Perform cross-validation or another approach (hint anova) in order to select the best degrees of freedom for a regression spline on this data. Describe your results.

Anova to compare models

```
spline.fit3 <- lm(nox ~ bs(dis,df = 3), data = df)
spline.fit4 <- lm(nox ~ bs(dis,df = 4), data = df)
spline.fit5 <- lm(nox ~ bs(dis,df = 5), data = df)
spline.fit6 <- lm(nox ~ bs(dis,df = 6), data = df)
spline.fit7 <- lm(nox ~ bs(dis,df = 7), data = df)
spline.fit8<- lm(nox ~ bs(dis,df = 8), data = df)
spline.fit9 <- lm(nox ~ bs(dis,df = 9), data = df)
spline.fit10 <- lm(nox ~ bs(dis,df = 10), data = df)
anova(spline.fit3,spline.fit4,spline.fit5,spline.fit6,spline.fit7,spline.fit8,spline.fit9,spline.fit10)
```

```
## Analysis of Variance Table
##
## Model 1: nox ~ bs(dis, df = 3)
## Model 2: nox ~ bs(dis, df = 4)
## Model 3: nox ~ bs(dis, df = 5)
## Model 4: nox ~ bs(dis, df = 6)
## Model 5: nox ~ bs(dis, df = 7)
## Model 6: nox ~ bs(dis, df = 8)
## Model 7: nox ~ bs(dis, df = 9)
## Model 8: nox ~ bs(dis, df = 10)
##   Res.Df    RSS Df Sum of Sq       F    Pr(>F)
## 1    502 1.9341
## 2    501 1.9228  1  0.011332  3.1292  0.077517 .
## 3    500 1.8402  1  0.082602 22.8102 2.359e-06 ***
## 4    499 1.8340  1  0.006207  1.7140  0.191074
## 5    498 1.8299  1  0.004081  1.1271  0.288918
## 6    497 1.8170  1  0.012889  3.5593  0.059796 .
## 7    496 1.8256  1 -0.008657
## 8    495 1.7925  1  0.033118  9.1453  0.002623 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The p-value for the comparison of model 1 and 2 is not significant so we could say that a 3rd degree spline is sufficient to explain the data, similar to what we found above!