Lab01 - Polynomial Regression

Solutions

This lab is adapted from Section 3.6 of ISLR.

R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com and https://www.rstudio.com/wp-content/uploads/2016/03/rmarkdown-cheatsheet-2.0.pdf.

When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. An R code chunk looks like this:

```
2 + 2
```

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

```
log(10)
```

```
## [1] 2.302585
```

There are three main ways to run R code. First, whenever you knit the document, all chunks will be run in a "fresh" R session.

However, as you're going along you will also want to run commands in a working session so that you can check that your code runs without having to knit the whole document. To do that, you can run individual code chunks by clicking the green "Play" arrow at the top right corner of the chunk.

You can also select individual lines of code you want to run and choose "Run... Run Selected Line(s)" from the menu at the top of the editor window.

The second two of these approaches will send commands to your Console, at the bottom of the screen. **Except for in times of desperation, you never want to enter commands directly into the Console!** Any commands you enter directly into the console will run one time only, and will not be a permanent part of your R Markdown document. **Always enter commands you want to save directly into your R Markdown document!!**.

Try out all three of those approaches with the example code chunk above.

Loading Packages

R comes with a decent amount of built-in functionality, but to do anything useful you will need to load *packages* that contain additional functionality. You load packages with the `library` command. Here we will load 3 packages with functionality you will need for this lab: readr, dplyr, and ggplot2. Run the code chunk below to load these packages.

```
library(readr)
library(dplyr)
```

```
## Warning: package 'dplyr' was built under R version 3.5.2
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 3.5.2
```

Boston Housing Data

The following R code reads in a data set with information about housing in 506 neighborhoods around Boston. It then uses the `head` function to look at the first few rows of the data.

```
Boston <- read_csv("http://www.evanlray.com/data/mass/Boston.csv")
```

```
## Parsed with column specification:
## cols(
##   crim = col_double(),
##   zn = col_double(),
##   indus = col_double(),
##   chas = col_double(),
##   nox = col_double(),
##   rm = col_double(),
##   age = col_double(),
##   dis = col_double(),
##   rad = col_double(),
##   tax = col_double(),
##   ptratio = col_double(),
##   black = col_double(),
##   lstat = col_double(),
##   medv = col_double()
## )
```

```
head(Boston)
```

```
## # A tibble: 6 x 14
##      crim    zn indus  chas   nox    rm   age   dis   rad   tax ptratio
##     <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>   <dbl>
## 1 0.00632    18  2.31     0 0.538  6.58  65.2  4.09     1   296    15.3
## 2 0.0273      0  7.07     0 0.469  6.42  78.9  4.97     2   242    17.8
## 3 0.0273      0  7.07     0 0.469  7.18  61.1  4.97     2   242    17.8
## 4 0.0324      0  2.18     0 0.458  7.00  45.8  6.06     3   222    18.7
## 5 0.0690      0  2.18     0 0.458  7.15  54.2  6.06     3   222    18.7
## 6 0.0298      0  2.18     0 0.458  6.43  58.7  6.06     3   222    18.7
## # ... with 3 more variables: black <dbl>, lstat <dbl>, medv <dbl>
```

We can find out the number of rows (observational units) and columns (variables) in the data set with the `dim` command:

```
dim(Boston)
```

```
## [1] 506  14
```

We see that we have data for 506 observational units and 14 variables. Each observational unit in this data set is a neighborhood around Boston. In this lab, we'll look at just two of the 14 variables:

- `nox` is the nitric oxide concentration (parts per 10 million). High concentrations of nitric oxide are hazardous to human health.
- `dis` is the weighted distance to five Boston employment centers.

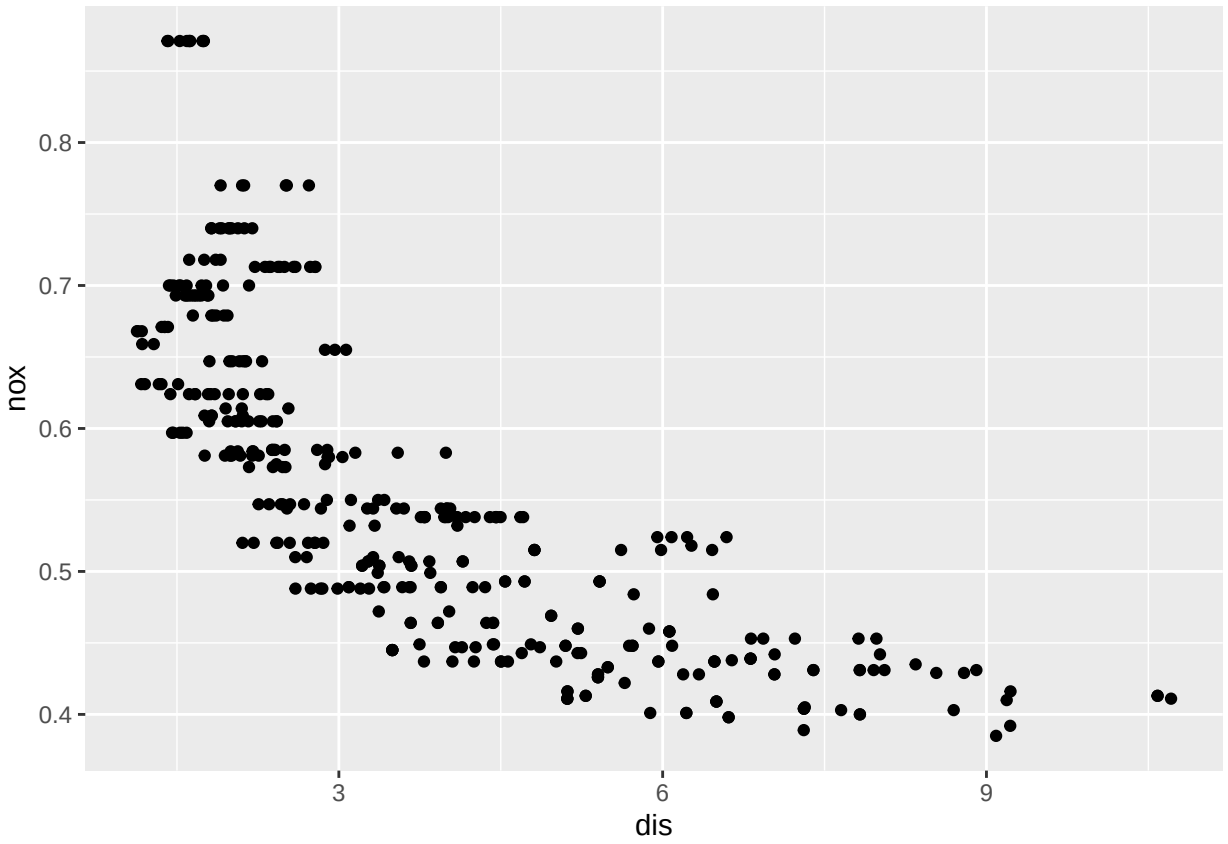
Train/Test Split

In this lab, you will fit several polynomial regression models to the data and compare their performance. Here, we set up a train/test split of the data, using 75% of the data for the training set and the remaining 25% for the test set. **Below, you will fit your models to the training data and evaluate and compare their predictive performance on the test data.**

```
set.seed(62585) # seed generated at random.org

num_train <- floor(0.75 * nrow(Boston))
train_inds <- sample(nrow(Boston), size = num_train) %>%
  sort()

train_Boston <- Boston %>%
  slice(train_inds)

test_Boston <- Boston %>%
  slice(-train_inds)
```

Problem 1: Make a scatter plot of the training set data using ggplot2. Put `dis` on the horizontal axis and `nox` on the vertical axis. (See the simple linear regression handout from the first day if you're not sure how to do this – we will discuss plots with ggplot2 in more detail in coming days.)

```
ggplot(data =  train_Boston, mapping = aes(x = dis, y = nox)) +
  geom_point()
```



Problem 2: Fit a polynomials of each degree 1 through 8 to the training set data, with `nox` as the response variable and `dis` as the explanatory variable. You will fit 8 separate models. Recall that you can use the `poly` function to avoid having to manually type out all the polynomial degree terms.

```
fit_deg1 <- lm(nox ~ dis, data = train_Boston)
fit_deg2 <- lm(nox ~ poly(dis, 2, raw = TRUE), data = train_Boston)
fit_deg3 <- lm(nox ~ poly(dis, 3, raw = TRUE), data = train_Boston)
fit_deg4 <- lm(nox ~ poly(dis, 4, raw = TRUE), data = train_Boston)
fit_deg5 <- lm(nox ~ poly(dis, 5, raw = TRUE), data = train_Boston)
fit_deg6 <- lm(nox ~ poly(dis, 6, raw = TRUE), data = train_Boston)
fit_deg7 <- lm(nox ~ poly(dis, 7, raw = TRUE), data = train_Boston)
fit_deg8 <- lm(nox ~ poly(dis, 8, raw = TRUE), data = train_Boston)
```

Problem 3: For each model fit, record the following things: (1) the training set MSE; (2) the test set MSE; (3) the highest polynomial degree that would be "statistically signficant" according to a hypothesis test conducted at the $\alpha = 0.05$ level.

```
summary(fit_deg1)
```

```
##
## Call:
```

```
## lm(formula = nox ~ dis, data = train_Boston)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.12057 -0.05117 -0.01248  0.04432  0.23065
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.715291   0.007689   93.03   <2e-16 ***
## dis         -0.042836   0.001803  -23.75   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.07168 on 377 degrees of freedom
## Multiple R-squared:  0.5995, Adjusted R-squared:  0.5984
## F-statistic: 564.3 on 1 and 377 DF,  p-value: < 2.2e-16
```

```
summary(fit_deg2)
```

```
##
## Call:
## lm(formula = nox ~ poly(dis, 2, raw = TRUE), data = train_Boston)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.117746 -0.044282 -0.009098  0.026996  0.203331
##
## Coefficients:
##                              Estimate Std. Error t value Pr(>|t|)
## (Intercept)                 0.8409499  0.0130943   64.22   <2e-16 ***
## poly(dis, 2, raw = TRUE)1 -0.1118002  0.0063779  -17.53   <2e-16 ***
## poly(dis, 2, raw = TRUE)2  0.0072873  0.0006533   11.15   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.06222 on 376 degrees of freedom
## Multiple R-squared:  0.6991, Adjusted R-squared:  0.6975
## F-statistic: 436.7 on 2 and 376 DF,  p-value: < 2.2e-16
```

```
summary(fit_deg3)
```

```
##
## Call:
## lm(formula = nox ~ poly(dis, 3, raw = TRUE), data = train_Boston)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.116255 -0.042161 -0.005435  0.024858  0.198273
##
## Coefficients:
##                              Estimate Std. Error t value Pr(>|t|)
## (Intercept)                 0.9144820  0.0255053  35.855  < 2e-16 ***
## poly(dis, 3, raw = TRUE)1 -0.1712467  0.0188591  -9.080  < 2e-16 ***
## poly(dis, 3, raw = TRUE)2  0.0203280  0.0039529   5.143 4.37e-07 ***
## poly(dis, 3, raw = TRUE)3 -0.0008195  0.0002451  -3.344  0.00091 ***
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.06139 on 375 degrees of freedom
## Multiple R-squared:  0.7078, Adjusted R-squared:  0.7054
## F-statistic: 302.8 on 3 and 375 DF,  p-value: < 2.2e-16
```

```
summary(fit_deg4)
```

```
##
## Call:
## lm(formula = nox ~ poly(dis, 4, raw = TRUE), data = train_Boston)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.118001 -0.040656 -0.005625  0.023663  0.198490
##
## Coefficients:
##                              Estimate Std. Error t value Pr(>|t|)
## (Intercept)                 8.747e-01  5.095e-02  17.167   <2e-16 ***
## poly(dis, 4, raw = TRUE)1 -1.285e-01  5.109e-02  -2.515   0.0123 *
## poly(dis, 4, raw = TRUE)2  5.622e-03  1.679e-02   0.335   0.7380
## poly(dis, 4, raw = TRUE)3  1.145e-03  2.194e-03   0.522   0.6021
## poly(dis, 4, raw = TRUE)4 -8.824e-05  9.793e-05  -0.901   0.3681
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.06141 on 374 degrees of freedom
## Multiple R-squared:  0.7084, Adjusted R-squared:  0.7053
## F-statistic: 227.2 on 4 and 374 DF,  p-value: < 2.2e-16
```

```
summary(fit_deg5)
```

```
##
## Call:
## lm(formula = nox ~ poly(dis, 5, raw = TRUE), data = train_Boston)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.125897 -0.040111 -0.007377  0.026720  0.194888
##
## Coefficients:
##                              Estimate Std. Error t value Pr(>|t|)
## (Intercept)                 6.136e-01  9.789e-02   6.268 1.01e-09 ***
## poly(dis, 5, raw = TRUE)1  2.284e-01  1.253e-01   1.823  0.06917 .
## poly(dis, 5, raw = TRUE)2 -1.657e-01  5.752e-02  -2.881  0.00419 **
## poly(dis, 5, raw = TRUE)3  3.776e-02  1.197e-02   3.156  0.00173 **
## poly(dis, 5, raw = TRUE)4 -3.630e-03  1.142e-03  -3.178  0.00161 **
## poly(dis, 5, raw = TRUE)5  1.260e-04  4.049e-05   3.112  0.00200 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.06071 on 373 degrees of freedom
## Multiple R-squared:  0.7158, Adjusted R-squared:  0.712
## F-statistic: 187.9 on 5 and 373 DF,  p-value: < 2.2e-16
```

```r
summary(fit_deg6)
```

```
##
## Call:
## lm(formula = nox ~ poly(dis, 6, raw = TRUE), data = train_Boston)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.131381 -0.038025 -0.009525  0.026908  0.188357
##
## Coefficients:
##                              Estimate Std. Error t value Pr(>|t|)
## (Intercept)                 1.384e-01  1.923e-01   0.720 0.472095
## poly(dis, 6, raw = TRUE)1  1.027e+00  3.056e-01   3.362 0.000853 ***
## poly(dis, 6, raw = TRUE)2 -6.670e-01  1.842e-01  -3.621 0.000334 ***
## poly(dis, 6, raw = TRUE)3  1.890e-01  5.417e-02   3.489 0.000542 ***
## poly(dis, 6, raw = TRUE)4 -2.705e-02  8.264e-03  -3.274 0.001160 **
## poly(dis, 6, raw = TRUE)5  1.914e-03  6.260e-04   3.057 0.002398 **
## poly(dis, 6, raw = TRUE)6 -5.309e-05  1.855e-05  -2.861 0.004455 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.06013 on 372 degrees of freedom
## Multiple R-squared:  0.7219, Adjusted R-squared:  0.7174
## F-statistic:   161 on 6 and 372 DF,  p-value: < 2.2e-16
```

```r
summary(fit_deg7)
```

```
##
## Call:
## lm(formula = nox ~ poly(dis, 7, raw = TRUE), data = train_Boston)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.132352 -0.038774 -0.005711  0.023601  0.193423
##
## Coefficients:
##                              Estimate Std. Error t value Pr(>|t|)
## (Intercept)                -6.200e-01  3.834e-01  -1.617 0.106728
## poly(dis, 7, raw = TRUE)1  2.537e+00  7.279e-01   3.485 0.000551 ***
## poly(dis, 7, raw = TRUE)2 -1.836e+00  5.441e-01  -3.375 0.000817 ***
## poly(dis, 7, raw = TRUE)3  6.483e-01  2.084e-01   3.112 0.002005 **
## poly(dis, 7, raw = TRUE)4 -1.268e-01  4.447e-02  -2.851 0.004601 **
## poly(dis, 7, raw = TRUE)5  1.400e-02  5.332e-03   2.625 0.009015 **
## poly(dis, 7, raw = TRUE)6 -8.161e-04  3.349e-04  -2.437 0.015275 *
## poly(dis, 7, raw = TRUE)7  1.951e-05  8.552e-06   2.282 0.023054 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.0598 on 371 degrees of freedom
## Multiple R-squared:  0.7258, Adjusted R-squared:  0.7206
## F-statistic: 140.3 on 7 and 371 DF,  p-value: < 2.2e-16
```

```r
summary(fit_deg8)
```

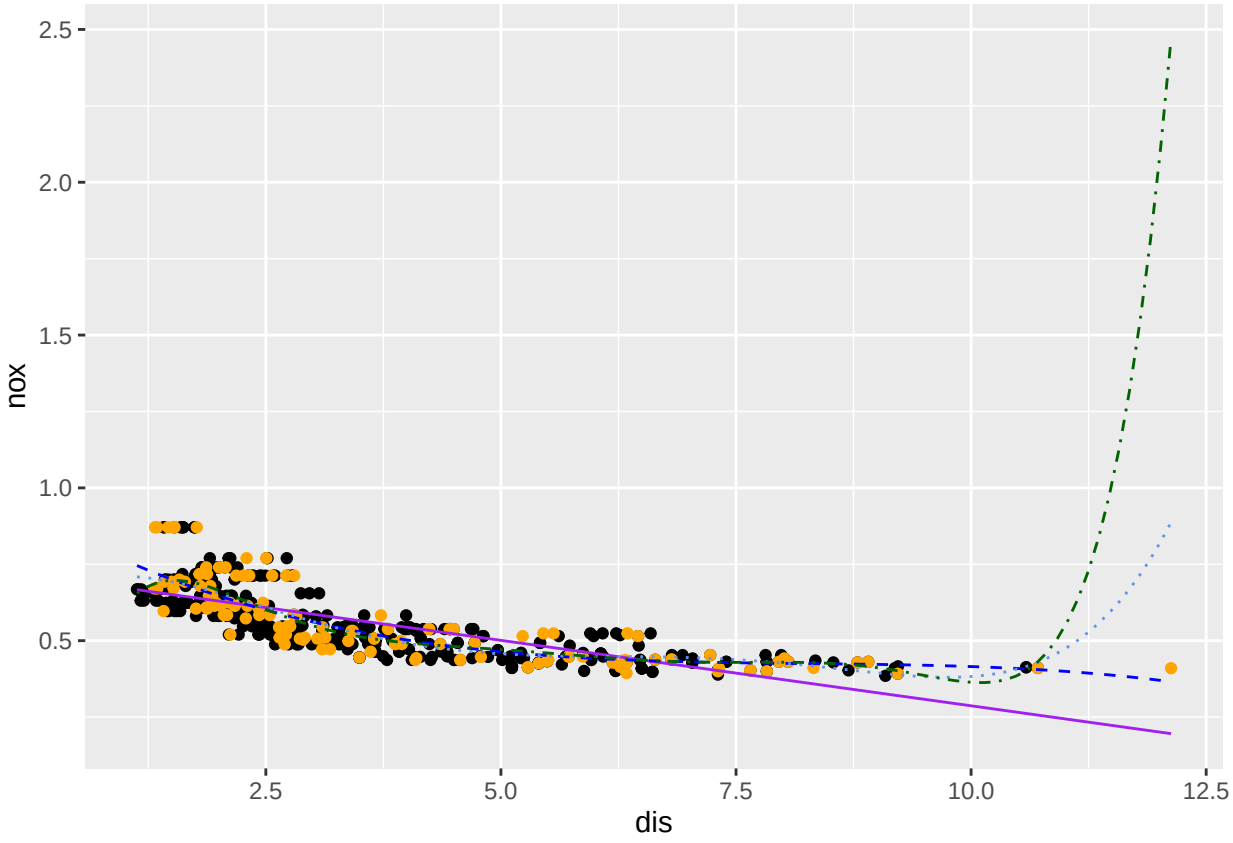
```
##
## Call:
## lm(formula = nox ~ poly(dis, 8, raw = TRUE), data = train_Boston)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.13095 -0.03994 -0.00516  0.02593  0.19575
##
## Coefficients:
##                               Estimate Std. Error t value Pr(>|t|)
## (Intercept)                 -1.341e+00  7.224e-01  -1.856  0.06428 .
## poly(dis, 8, raw = TRUE)1  4.196e+00  1.587e+00   2.644  0.00853 **
## poly(dis, 8, raw = TRUE)2 -3.373e+00  1.414e+00  -2.384  0.01761 *
## poly(dis, 8, raw = TRUE)3  1.398e+00  6.704e-01   2.086  0.03770 *
## poly(dis, 8, raw = TRUE)4 -3.391e-01  1.858e-01  -1.825  0.06883 .
## poly(dis, 8, raw = TRUE)5  4.997e-02  3.103e-02   1.610  0.10819
## poly(dis, 8, raw = TRUE)6 -4.403e-03  3.067e-03  -1.436  0.15191
## poly(dis, 8, raw = TRUE)7  2.132e-04  1.648e-04   1.294  0.19664
## poly(dis, 8, raw = TRUE)8 -4.360e-06  3.705e-06  -1.177  0.24007
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.05976 on 370 degrees of freedom
## Multiple R-squared:  0.7268, Adjusted R-squared:  0.7209
## F-statistic:   123 on 8 and 370 DF,  p-value: < 2.2e-16
```

```
results <- data.frame(
  poly_degree = 1:8,
  train_mse = NA,
  test_mse = NA,
  highest_sig_degree = c(1, 2, 3, 1, 5, 6, 7, 3)
)

for(degree in 1:8) {
  fit <- get(paste0("fit_deg", degree))
  results$train_mse[degree] <-
    mean((train_Boston$nox - predict(fit))^2)
  results$test_mse[degree] <-
    mean((test_Boston$nox - predict(fit, newdata = test_Boston))^2)
}

results
```

```
##   poly_degree   train_mse    test_mse highest_sig_degree
## 1           1 0.005111551 0.006565464                  1
## 2           2 0.003840718 0.004599455                  2
## 3           3 0.003729517 0.004146975                  3
## 4           4 0.003721438 0.004322765                  1
## 5           5 0.003627282 0.006021153                  5
## 6           6 0.003549161 0.010616504                  6
## 7           7 0.003500033 0.037503357                  7
## 8           8 0.003486983 0.033063707                  3
```

Problem 4: Make a scatter plot showing the training set data in one color and the test set data in a second color, and curves showing your model fits of degree 1, 3, 5, and 7. You might use a different color and linetype to distinguish the different model fits.

```
predict_deg1 <- function(x) {
  predict(fit_deg1, data.frame(dis = x))
}
predict_deg3 <- function(x) {
  predict(fit_deg3, data.frame(dis = x))
}
predict_deg5 <- function(x) {
  predict(fit_deg5, data.frame(dis = x))
}
predict_deg7 <- function(x) {
  predict(fit_deg7, data.frame(dis = x))
}

ggplot(data =  train_Boston, mapping = aes(x = dis, y = nox)) +
  geom_point() +
  geom_point(data = test_Boston, mapping = aes(x = dis, y = nox), color = "orange") +
  stat_function(fun = predict_deg1, color = "purple") +
  stat_function(fun = predict_deg3, color = "blue", linetype = 2) +
  stat_function(fun = predict_deg5, color = "cornflowerblue", linetype = 3) +
  stat_function(fun = predict_deg7, color = "darkgreen", linetype = 4)
```



Problem 5: Discuss your answers to Problems 3 and 4. Here are some things to think about:

(a) Which model or models seem best based on your plots?

Out of the models included in the plot, the degree 3 fit seems best. The degree 1 fit does not capture the non-linearity in the data, and the degree 5 and 7 polynomials have too much extreme curvature on the right hand side of the plot.

(b) Which model or models seem best based on test set MSE? Is this result consistent which what you see in the plots?

The test set MSE is lowest for the degree 3 polynomial. All of the degree 2, 3, and 4 models have similar MSE. This is consistent with the conclusion from the plot.

(c) If you had used training set MSE to select the model, would you have made a good decision?

The training set MSE was lowest for the degree 8 polynomial fit. This is not the best choice.

Training set MSE is not a reliable measure to use for selecting the polynomial degree.

(d) Do the hypothesis tests tell a consistent story about what polynomial degree to choose? If you had just started with a degree 7 polynomial and decided on the polynomial degree based on the hypothesis tests from that model, would you have made a good decision?

The hypothesis test results are very inconsistent. For example, the test results for the degree 3 and degree 5 polynomials both indicate that all of the terms included in those models are statistically significant and should be included. However, the test results for the degree 4 polynomial suggest that not all of those terms are contributing to the model fit.

If we started with a degree 7 polynomial model, the hypothesis tests would have indicated we should keep all of those polynomial terms in the model. However, we have seen from the plots and test set evaluation above that the degree 7 fit is not the best model for these data.

Hypothesis test results are not a reliable measure to use for selecting the polynomial degree.

Problem 6: Just for kicks, try fitting a polynomial of degree 13 to the data. Print out the summary for this model fit. What problems does it indicate? Now fit the model again, but specify raw = FALSE to the poly function. Does this solve the problem? We will discuss what's going on next class.

```
fit_deg13a <- lm(nox ~ poly(dis, 13, raw = TRUE), data = train_Boston)
summary(fit_deg13a)
```

```
##
## Call:
## lm(formula = nox ~ poly(dis, 13, raw = TRUE), data = train_Boston)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.130355 -0.038377 -0.006855  0.028114  0.195666
##
## Coefficients: (1 not defined because of singularities)
```

```
##                               Estimate Std. Error t value Pr(>|t|)
## (Intercept)                 -9.172e+00  1.071e+01  -0.857    0.392
## poly(dis, 13, raw = TRUE)1   3.150e+01  3.625e+01   0.869    0.386
## poly(dis, 13, raw = TRUE)2  -4.466e+01  5.315e+01  -0.840    0.401
## poly(dis, 13, raw = TRUE)3   3.728e+01  4.472e+01   0.834    0.405
## poly(dis, 13, raw = TRUE)4  -2.033e+01  2.411e+01  -0.843    0.400
## poly(dis, 13, raw = TRUE)5   7.588e+00  8.795e+00   0.863    0.389
## poly(dis, 13, raw = TRUE)6  -1.981e+00  2.234e+00  -0.887    0.376
## poly(dis, 13, raw = TRUE)7   3.643e-01  3.993e-01   0.912    0.362
## poly(dis, 13, raw = TRUE)8  -4.688e-02  5.000e-02  -0.937    0.349
## poly(dis, 13, raw = TRUE)9   4.124e-03  4.290e-03   0.961    0.337
## poly(dis, 13, raw = TRUE)10 -2.360e-04  2.400e-04  -0.984    0.326
## poly(dis, 13, raw = TRUE)11  7.907e-06  7.878e-06   1.004    0.316
## poly(dis, 13, raw = TRUE)12 -1.175e-07  1.151e-07  -1.021    0.308
## poly(dis, 13, raw = TRUE)13         NA         NA      NA       NA
##
## Residual standard error: 0.05995 on 366 degrees of freedom
## Multiple R-squared:  0.7281, Adjusted R-squared:  0.7192
## F-statistic: 81.66 on 12 and 366 DF,  p-value: < 2.2e-16
```

```
fit_deg13b <- lm(nox ~ poly(dis, 13, raw = FALSE), data = train_Boston)
summary(fit_deg13b)
```

```
##
## Call:
## lm(formula = nox ~ poly(dis, 13, raw = FALSE), data = train_Boston)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.13052 -0.03835 -0.00629  0.02842  0.19581
##
## Coefficients:
##                               Estimate Std. Error t value Pr(>|t|)
## (Intercept)                   0.554939   0.003083 179.971  < 2e-16 ***
## poly(dis, 13, raw = FALSE)1  -1.702874   0.060029 -28.367  < 2e-16 ***
## poly(dis, 13, raw = FALSE)2   0.694007   0.060029  11.561  < 2e-16 ***
## poly(dis, 13, raw = FALSE)3  -0.205292   0.060029  -3.420 0.000697 ***
## poly(dis, 13, raw = FALSE)4  -0.055334   0.060029  -0.922 0.357251
## poly(dis, 13, raw = FALSE)5   0.188906   0.060029   3.147 0.001786 **
## poly(dis, 13, raw = FALSE)6  -0.172069   0.060029  -2.866 0.004392 **
## poly(dis, 13, raw = FALSE)7   0.136455   0.060029   2.273 0.023598 *
## poly(dis, 13, raw = FALSE)8  -0.070325   0.060029  -1.172 0.242156
## poly(dis, 13, raw = FALSE)9   0.036841   0.060029   0.614 0.539780
## poly(dis, 13, raw = FALSE)10 -0.003106   0.060029  -0.052 0.958759
## poly(dis, 13, raw = FALSE)11 -0.033682   0.060029  -0.561 0.575081
## poly(dis, 13, raw = FALSE)12 -0.061235   0.060029  -1.020 0.308361
## poly(dis, 13, raw = FALSE)13  0.006343   0.060029   0.106 0.915912
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.06003 on 365 degrees of freedom
## Multiple R-squared:  0.7281, Adjusted R-squared:  0.7184
## F-statistic: 75.18 on 13 and 365 DF,  p-value: < 2.2e-16
```

The first fit indicates a problem with a design matrix that is not full rank. This is not an issue in the second

fit.