MSSC 6250 Machine Learning Homework 1

Bias-Variance Tradeoff and Linear Regression

Henri Medeiros Dos Reis

- Deadline: **Friday, Feb 3 23:59 PM**
- Homework presentation date: **Tuesday, Feb 7**
- Please submit your work in **one PDF** file to **D2L > Assessments > Dropbox**. *Multiple files or a file that is not in pdf format are not allowed.*
- Any relevant code should be attached.

Exercises required for all students

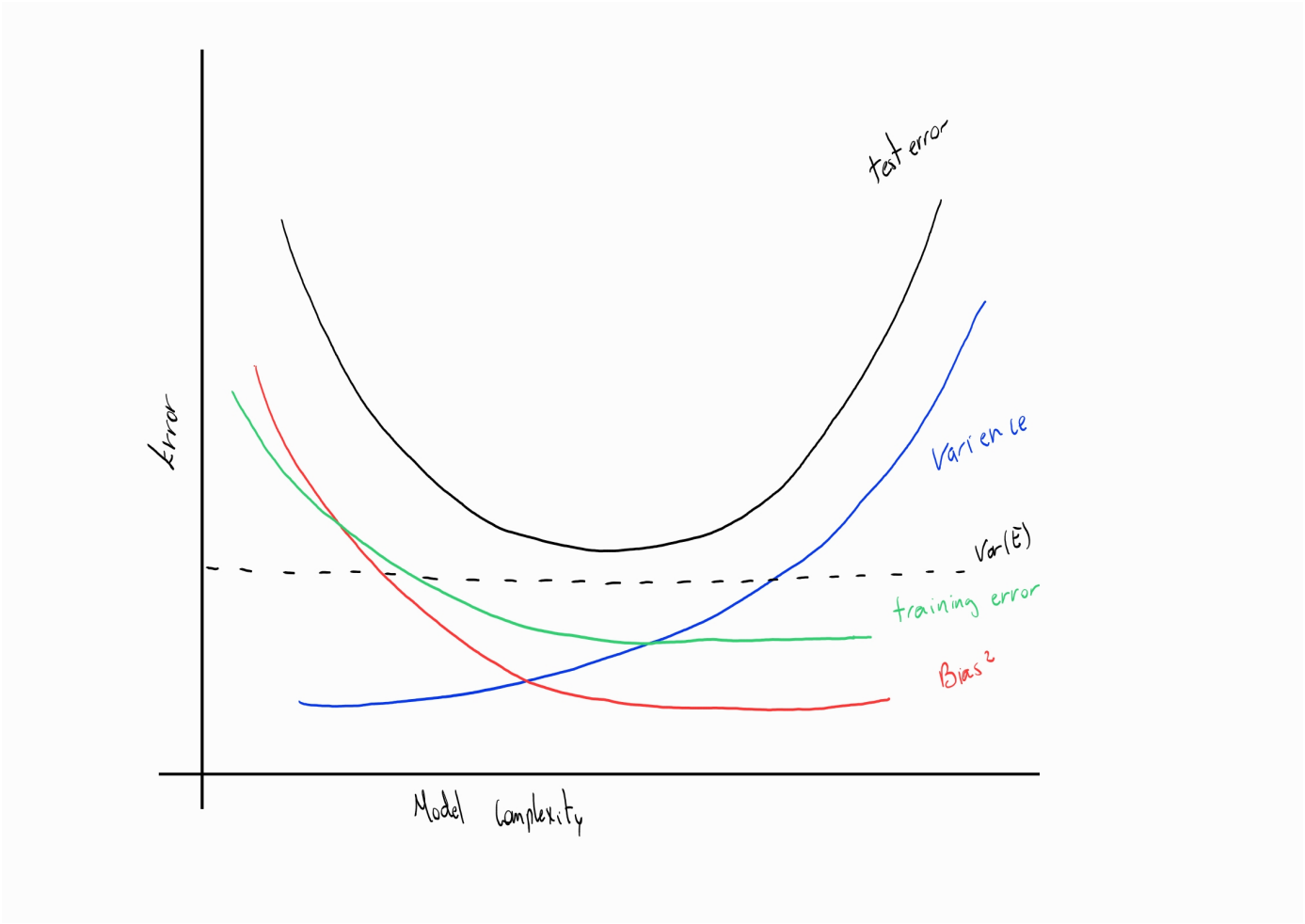
- Read **ISL** Chapter 2 and 3.
- **ISL** Sec. 2.4: 1, 3, 5, 6

Solution 2.4.1:

- *a):* The flexible model will perform better. Since we have a large sample size, the flexible model is going to fit the model better
- *b):* A flexible method would perform worse on this case. Since the flexible model is likely going to overfit due the small sample size.
- *c):* A flexible method would perform better. Since we have non-linearity, the number of degrees of freedom is important, the higher the better to capture the non-linearity of the data.
- *d):* A flexible method would perform worse. Because a flexible model is likely to take the random errors into account way more when fitting the data, variance of the model would also increase.

Solution 2.4.3:

- *a):*



- *b):* Var(E) is error that can not be reduced, and no test predictions can perform better than this, and because of this, it is a straight line. Test error reduces to the best solution point as increased flexibility, which means a better fit, with if we increase it by any amount, it will lead to overfitting our model. Training error decreases continuously, which means that the error is decreasing as we increase the flexibility of the model, then this model will very closely fit the training data. Variance increases as the method tends to overfit as flexibility increases, which means that we are taking training more than needed, and our model is going to perform worse in testing. Bias squared is going to reduce as the flexibility goes up, since the model is better at fitting the data.

Solution 2.4.5:

Using a flexible approach for regression or classification is likely going to give you a better fit for any non-linear models. However, we will need to use more regressors in our model, which can

lead to overfitting. Then decreasing bias while paying the price by increasing variance. On the other hand, using a less flexible approach is going to make the interpretability of the results way easier, since you will have to analyze less predictors and possibly less transformations.

Solution 2.4.6:

Using a parametric approaches works better when we know the form of the function f, and f is linear since it assumes what f looks like. Leading to simpler models and more interpretable results, since it uses less parameters.

While a non-parametric is a better choice when we do not know or are not comfortable assuming the shape of f, it leads to a more complex model, but since most of the interesting problems that we have to solve are complex, we are good with that. It definitely makes the results harder to interpret, but the results are going to be more accurate.

A non-parametric approach does not assume a functional form for f and so requires a very large number of observations to accurately estimate f.

The advantages of a parametric approach to regression or classification are the simplifying of modeling f to a few parameters and not as many observations are required compared to a non-parametric approach.

The disadvantages of a parametric approach to regression or classification are a potential to inaccurately estimate f if the form of f assumed is wrong or to overfit the observations if more flexible models are used.

- **ISL** Sec. 3.7: 3, 4, 5, 6, 7, 11, 13

Solution 3.7.3:

- *a):* iii is True. Our model is of the form:

  ```
  Y = 50 + 20*GPA + 0.07*IQ + 35*Level+ 0.01*GPA:IQ - 10*GPA:Level.
  ```

  Therefore, we can see that if the GPA goes higher than 3.5, high school graduates earn more than college graduates.

- *b):*

```
pred = 50 + 20*4 + 0.07*110 + 35*1+ 0.01*(4*110) - 10*(4*1)
cat("the salary will be:",pred,"k")
```

```
the salary will be: 137.1 k
```

- *c):*

  False, it may seem small, but it may be multiplied with big numbers. To determine if it is statistically significant or not, we would need further analysis.

Solution 3.7.4:

- *a):* Even though we are using a higher polynomial than needed, adding extra predictors is always going to decrease the value of RSS, because we will be overfitting the data. So the cubic is going to perform better in training.
- *b):* Now for testing data, the story is a little different, since we overfitted our model with the cubic, the training scores will decrease. Making the linear regression better, therefore just a simple linear regression to have lower than the cubic one
- *c):* Higher degree polynomials will always fit better when talking about non-linear relationships, therefore, the cubic one will have a smaller RSS, even more in training.
- *d):* It is not possible to tell from the given information. The results will highly depend on the function f, if it is close to being linear, the linear one will be better, but if it is closer to cubic, the cubic one will be better.

Solution 3.7.5:

$$\hat{y}_i = x_i\hat{\beta}, \text{ where } \hat{\beta} = \frac{\sum_{j=1}^{n} x_j y_j}{\sum_{k=1}^{n} x_k^2}$$

$$\hat{y}_i = x_i \frac{\sum_{j=1}^{n} x_j y_j}{\sum_{k=1}^{n} x_k^2} = \sum_{j=1}^{n} \frac{x_i x_j}{\sum_{k=1}^{n} x_k^2} y_j$$

Then:

$$a_j = \frac{x_i x_j}{\sum_{k=1}^{n} x_k^2} \Rightarrow \hat{y}_i = \sum_{j=1}^{n} a_j y_j$$

Solution 3.7.6:

$$\hat{\beta}_1 = \frac{\sum_{i=1}^{n}(x_i - \hat{x})(y_i - \hat{y})}{\sum_{i=1}^{n}(x_i - \hat{x}^2)}, \hat{\beta}_0 = \hat{y} - \hat{\beta}_1\hat{x}, y = \beta_0 + \beta_1 x$$

And we can use the fact that the right hand side should be equal to 0 if $(\hat{x}, \hat{y})$ is a point on the line. Therefore

$$0 = \hat{\beta}_0 + \hat{\beta}_1\hat{x} - \hat{y} \Rightarrow 0 = \hat{y} - \hat{\beta}_1\hat{x} + \hat{\beta}_1\hat{x} - \hat{y} = 0$$

Meaning that a line will always pass through $(\hat{x}, \hat{y})$.

Solution 3.7.7:

$$R^2 = 1 - \frac{RSS}{TSS}$$

$$Cor(X, Y) = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}} = \frac{\sum_{i=1}^{n}(x_i)(y_i)}{\sqrt{\sum_{i=1}^{n}(x_i)^2}\sqrt{\sum_{i=1}^{n}(y_i)^2}}$$

Since $\bar{x} = \bar{y} = 0$. And we want to show that $R^2 = (Cor(X,Y))^2$. Also, $RSS = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2$, $TSS = \sum_{i=1}^{n} y_i^2$, and $\hat{y}_i = \hat{\beta}_1 x_i$. Then

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n} y_i^2} = \frac{\sum_{i=1}^{n} y_i^2 - \sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n} y_i^2} = \frac{\sum_{i=1}^{n} y_i^2 - (y_i^2 - 2y_i\hat{y}_i + \hat{y}_i{}^2)}{\sum_{i=1}^{n} y_i^2}$$

$$\Rightarrow \frac{\sum_{i=1}^{n}(y_i^2 - y_i^2 + 2y_i\hat{y}_i - \hat{y}_i{}^2)}{\sum_{i=1}^{n} y_i^2} = \frac{\sum_{i=1}^{n} 2y_i\hat{y}_i - \hat{y}_i{}^2}{\sum_{i=1}^{n} y_i^2} = \frac{\sum_{i=1}^{n}(2y_i\hat{\beta}_1 x_i - (\hat{\beta}_1 x_i)^2)}{\sum_{i=1}^{n} y_i^2}$$

And we need to substitute $\hat{\beta}_1$, but let's simplify it first.

$$\hat{\beta}_1 = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n}(x_i - \bar{x})^2} = \frac{\sum_{i=1}^{n} x_i y_i}{\sum_{i=1}^{n} x_i^2}$$

Then

$$\Rightarrow \frac{\sum_{i=1}^{n}(2y_i \frac{\sum_{i=1}^{n} x_i y_i}{\sum_{i=1}^{n} x_i^2} x_i - (\frac{\sum_{i=1}^{n} x_i y_i}{\sum_{i=1}^{n} x_i^2} x_i)^2)}{\sum_{i=1}^{n} y_i^2} = \frac{2\frac{\sum_{i=1}^{n}(x_i y_i)^2}{\sum_{i=1}^{n} x_i^2} - \frac{\sum_{i=1}^{n}(x_i y_i)^2}{\sum_{i=1}^{n} x_i^2}}{\sum_{i=1}^{n} y_i^2}$$

$$\Rightarrow \frac{\frac{\sum_{i=1}^{n}(x_i y_i)^2}{\sum_{i=1}^{n} x_i^2}}{\sum_{i=1}^{n} y_i^2} = \frac{\sum_{i=1}^{n}(x_i y_i)^2}{\sum_{i=1}^{n} y_i^2 \sum_{i=1}^{n} x_i^2} = Cor(X,Y)^2$$

Solution 3.7.11:

- *a):*

```
set.seed (1)
x <- rnorm (100)
y <- 2 * x + rnorm (100)

model1 = lm(y~x+0)
summary(model1)
```

```
Call:
lm(formula = y ~ x + 0)

Residuals:
    Min      1Q  Median      3Q     Max
-1.9154 -0.6472 -0.1771  0.5056  2.3109

Coefficients:
```

```
  Estimate Std. Error t value Pr(>|t|)
x   1.9939     0.1065   18.73   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.9586 on 99 degrees of freedom
Multiple R-squared:  0.7798,    Adjusted R-squared:  0.7776
F-statistic: 350.7 on 1 and 99 DF,  p-value: < 2.2e-16
```

The coefficient is 1.99, the Std. Error is 0.11, t-statistic is 18.73 and the p-value is 0. Looking at the results we can see that the null hypothesis can be rejected, because the coefficient is statistically significant using $\alpha = 0.05$.

- *b):*

```r
model2 = lm(x~y+0)
summary(model2)
```

```
Call:
lm(formula = x ~ y + 0)

Residuals:
    Min      1Q  Median      3Q     Max
-0.8699 -0.2368  0.1030  0.2858  0.8938

Coefficients:
  Estimate Std. Error t value Pr(>|t|)
y  0.39111    0.02089   18.73   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.4246 on 99 degrees of freedom
Multiple R-squared:  0.7798,    Adjusted R-squared:  0.7776
F-statistic: 350.7 on 1 and 99 DF,  p-value: < 2.2e-16
```

The coefficient is 0.399, the Std. Error is 0.02, t-statistic is 18.73 and the p-value is 0. Looking at the results we can see that the null hypothesis can be rejected, because the coefficient is statistically significant using $\alpha = 0.05$. Also, the results for p-value and t-statistic are the same for both.

- *c):* As said before, we can see that the value of t-statistic, and p-values are the same for y onto x and x onto y regression.
- *d):*

$$t_{test} = \frac{\beta}{se(\beta)} = \frac{\beta}{\sqrt{\frac{\sum_{i=1}^{n}(y_i - x_i\beta)^2}{(n-1)\sum_{i=1}^{n} x_i^2}}}$$

$$\Rightarrow t_{test}^2 = \frac{\beta^2}{\frac{\sum_{i=1}^{n}(y_i - x_i\beta)^2}{(n-1)\sum_{i=1}^{n} x_i^2}}$$

And we know $\beta = \frac{\sum_{i=1}^{n} x_i y_i}{\sum_{i=1}^{n} x_i^2}$, then

$$\Rightarrow t_{test}^2 = \frac{(\frac{\sum_{i=1}^{n} x_i y_i}{\sum_{i=1}^{n} x_i^2})^2}{\frac{\sum_{i=1}^{n}(y_i - x_i(\frac{\sum_{i=1}^{n} x_i y_i}{\sum_{i=1}^{n} x_i^2}))^2}{(n-1)\sum_{i=1}^{n} x_i^2}} = (\frac{\sum_{i=1}^{n} x_i y_i}{\sum_{i=1}^{n} x_i^2})^2 \frac{(n-1)\sum_{i=1}^{n} x_i^2}{\sum_{i=1}^{n}(y_i - x_i(\frac{\sum_{i=1}^{n} x_i y_i}{\sum_{i=1}^{n} x_i^2}))^2}$$

$$\Rightarrow t_{test}^2 = \frac{(n-1)\sum_{i=1}^{n}(x_i y_i)^2}{\sum_{i=1}^{n} x_i^2 \sum_{i=1}^{n}(y_i - x_i\frac{\sum_{i=1}^{n} x_i y_i}{\sum_{i=1}^{n} x_i^2})^2} = \frac{(n-1)\sum_{i=1}^{n}(x_i y_i)^2}{\sum_{i=1}^{n} x_i^2 \sum_{i=1}^{n}(y_i^2 - 2\frac{\sum_{i=1}^{n} x_i y_i}{\sum_{i=1}^{n} x_i^2} x_i y_i + x_i^2(\frac{\sum_{i=1}^{n} x_i y_i}{\sum_{i=1}^{n} x_i^2})^2)}$$

$$\Rightarrow \frac{(n-1)\sum_{i=1}^{n}(x_i y_i)^2}{\sum_{i=1}^{n} x_i^2 \sum_{i=1}^{n}(y_i^2 - 2\frac{\sum_{i=1}^{n} x_i^2 y_i^2}{\sum_{i=1}^{n} x_i^2} + x_i^2(\frac{\sum_{i=1}^{n} x_i y_i}{\sum_{i=1}^{n} x_i^2})^2)} = \frac{(n-1)\sum_{i=1}^{n}(x_i y_i)^2}{\sum_{i=1}^{n} x_i^2 \sum_{i=1}^{n} y_i^2 - 2\sum_{i=1}^{n} x_i^2 y_i^2 + \sum_{i=1}^{n} x_i^2 y_i^2}$$

$$\Rightarrow \frac{(n-1)\sum_{i=1}^{n}(x_i y_i)^2}{\sum_{i=1}^{n} x_i^2 \sum_{i=1}^{n} y_i^2 - \sum_{i=1}^{n} x_i^2 y_i^2}$$

Finally:

$$t_{test} = \sqrt{\frac{(n-1)\sum_{i=1}^{n}(x_i y_i)^2}{\sum_{i=1}^{n} x_i^2 \sum_{i=1}^{n} y_i^2 - \sum_{i=1}^{n} x_i^2 y_i^2}} = \frac{\sqrt{(n-1)}\sum_{i=1}^{n}(x_i y_i)}{\sqrt{\sum_{i=1}^{n} x_i^2 \sum_{i=1}^{n} y_i^2 - \sum_{i=1}^{n} x_i^2 y_i^2}}$$

```
top = sqrt(length(x)-1)*sum(x*y)
bot = sqrt(sum(x2)*sum(y2)-sum(x*y)2)
print(top/bot)
```

```
[1] 18.72593
```

- *e):* We can see that exchanging y_i and x_i does not change anything, since in all the occasions they appear, it is a product.
- *f):*

```
mod_with_int <- lm(x~y)
summary(mod_with_int)
```

```
Call:
lm(formula = x ~ y)

Residuals:
     Min       1Q   Median       3Q      Max
-0.90848 -0.28101  0.06274  0.24570  0.85736

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.03880    0.04266    0.91    0.365
y            0.38942    0.02099   18.56   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.4249 on 98 degrees of freedom
Multiple R-squared:  0.7784,    Adjusted R-squared:  0.7762
F-statistic: 344.3 on 1 and 98 DF,  p-value: < 2.2e-16
```

```r
mod_with_int2 <- lm(y~x)
summary(mod_with_int2)
```

```
Call:
lm(formula = y ~ x)

Residuals:
    Min      1Q  Median      3Q     Max
-1.8768 -0.6138 -0.1395  0.5394  2.3462

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -0.03769    0.09699  -0.389    0.698
x            1.99894    0.10773  18.556   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.9628 on 98 degrees of freedom
Multiple R-squared:  0.7784,    Adjusted R-squared:  0.7762
F-statistic: 344.3 on 1 and 98 DF,  p-value: < 2.2e-16
```

As we can see, both t-statistic are 18.5556.

Solution 3.7.13:

- *a):*

```r
set.seed(1)
x = rnorm(100,0,1)
```

- *b):*

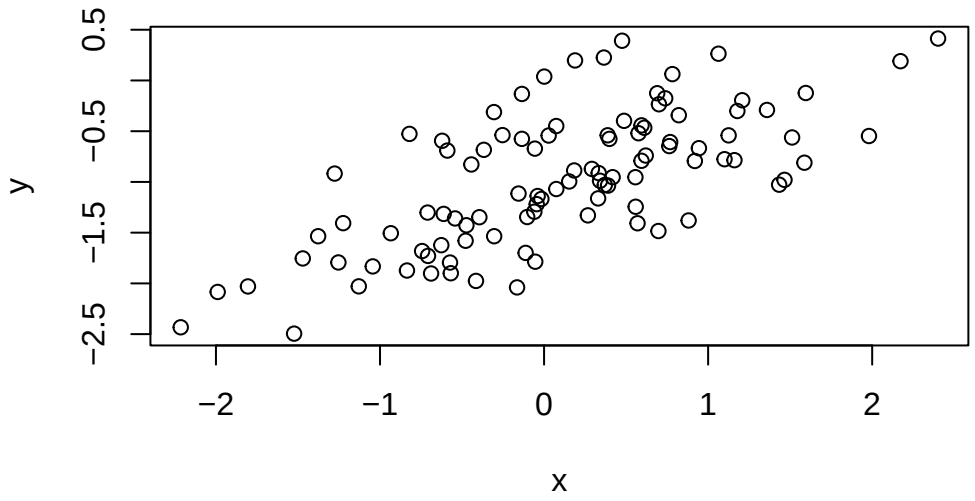
```r
eps = rnorm(100, 0, sqrt(0.25))
```

- *c):*

```r
y = -1 + 0.5*x + eps
```

- *d):*

```r
plot(x,y)
```



We can see that there is a positive linear relationship between x and y.

- *e):*

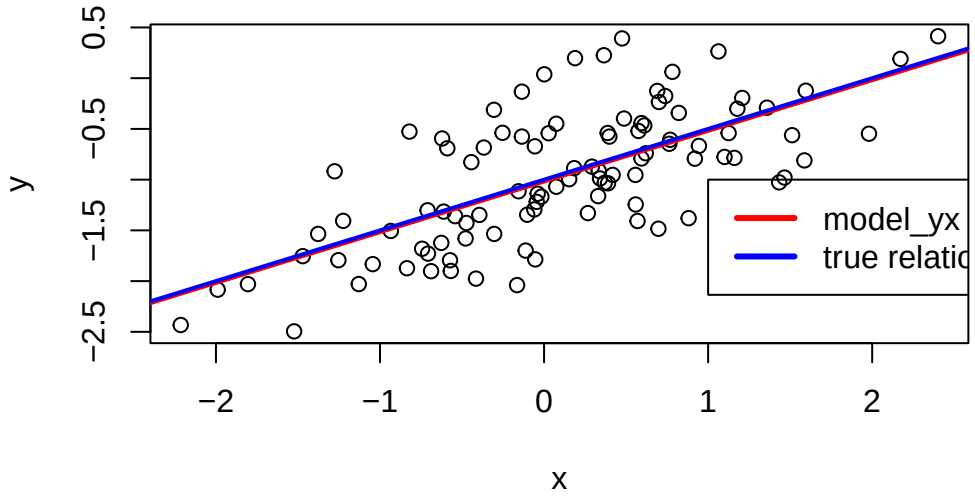
```r
model_yx = lm(y~x)
summary(model_yx)
```

```
Call:
lm(formula = y ~ x)

Residuals:
     Min       1Q   Median       3Q      Max
-0.93842 -0.30688 -0.06975  0.26970  1.17309

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -1.01885    0.04849 -21.010  < 2e-16 ***
x            0.49947    0.05386   9.273 4.58e-15 ***
---
Signif. codes:  0 ’***’ 0.001 ’**’ 0.01 ’*’ 0.05 ’.’ 0.1 ’ ’ 1

Residual standard error: 0.4814 on 98 degrees of freedom
Multiple R-squared:  0.4674,    Adjusted R-squared:  0.4619
F-statistic: 85.99 on 1 and 98 DF,  p-value: 4.583e-15
```

Our regression estimates are pretty close to the true values that we declared before . P-values are near zero and F-statistic is large, therefore we reject the null hypothesis, and our results are significant.

- *f):*

```r
plot(x, y)
abline(model_yx, lwd=2, col="red")
abline(-1, 0.5, lwd=2, col="blue")
legend(-1, legend = c("model_yx", "true relationship"), col=c("red","blue"), lwd=
```



- *g):*

```
model_sq = lm(y~x+I(x2))
summary(model_sq)
```

```
Call:
lm(formula = y ~ x + I(x^2))

Residuals:
     Min       1Q   Median       3Q      Max
-0.98252 -0.31270 -0.06441  0.29014  1.13500

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -0.97164    0.05883 -16.517  < 2e-16 ***
x            0.50858    0.05399   9.420  2.4e-15 ***
I(x^2)      -0.05946    0.04238  -1.403    0.164
---
Signif. codes:  0 ’***’ 0.001 ’**’ 0.01 ’*’ 0.05 ’.’ 0.1 ’ ’ 1

Residual standard error: 0.479 on 97 degrees of freedom
Multiple R-squared:  0.4779,    Adjusted R-squared:  0.4672
F-statistic:  44.4 on 2 and 97 DF,  p-value: 2.038e-14
```

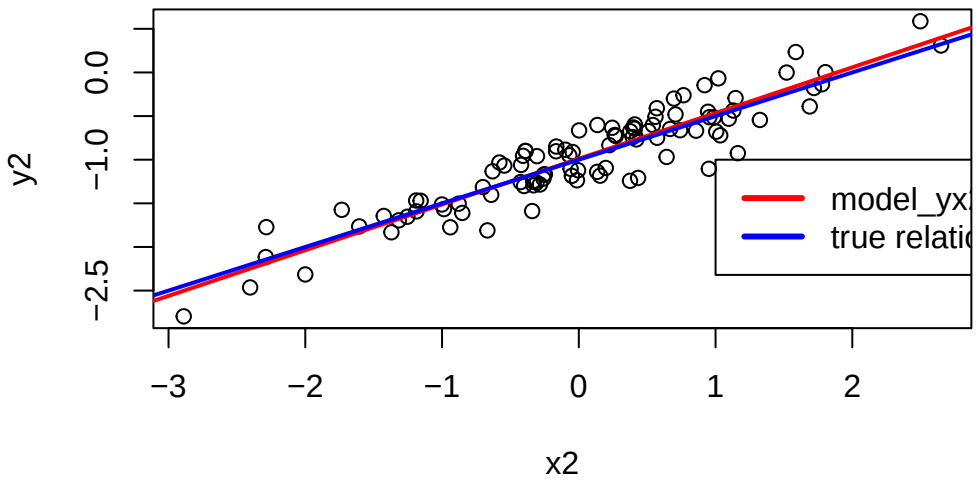
We can see that the model increased it's performance with the training data by a little. However, when looking at the p-value and t-statistic for x^2 we can see that the relationship is not statistically significant, and there is not a strong enough relationship between y and x^2.

- *h):*

```
x2 = rnorm(100)
eps2 = rnorm(100, 0, sqrt(0.05))
y2 = -1 + 0.5*x2 + eps2
plot(x2, y2)
model_yx2 = lm(y2~x2)
summary(model_yx2)
```

```
Call:
lm(formula = y2 ~ x2)

Residuals:
     Min       1Q   Median       3Q      Max
-0.61308 -0.12553 -0.00391  0.15199  0.41332

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -0.98917    0.02216  -44.64   <2e-16 ***
x2           0.52375    0.02152   24.33   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.2215 on 98 degrees of freedom
Multiple R-squared:  0.858, Adjusted R-squared:  0.8565
F-statistic: 592.1 on 1 and 98 DF,  p-value: < 2.2e-16
```

```
abline(model_yx2, lwd=2, col="red")
abline(-1, 0.5, lwd=2, col="blue")
legend(-1, legend = c("model_yx2", "true relationship"), col=c("red","blue"), lwd
```



As expected, the model performance improved, since there is less noise.

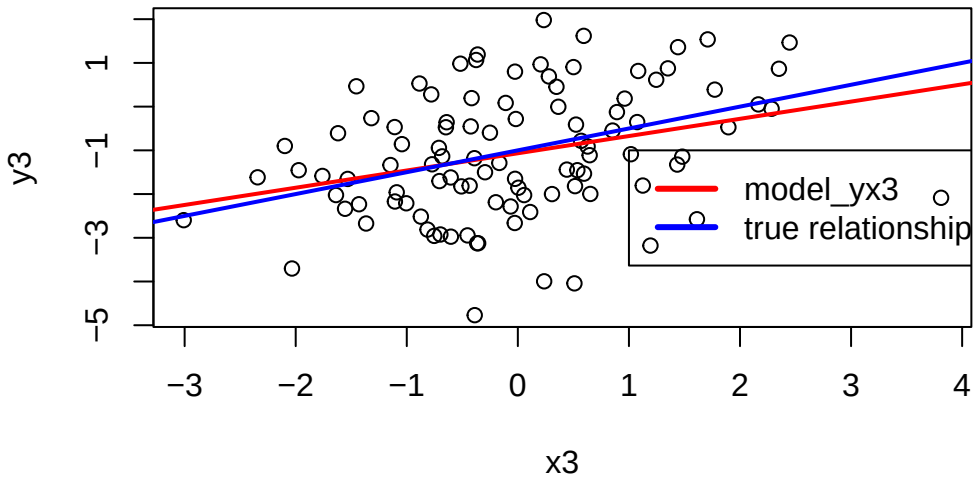
- *i):*

```
x3 = rnorm(100)
eps3 = rnorm(100, 0, sqrt(2))
y3 = -1 + 0.5*x3 + eps3
plot(x3, y3)
model_yx3 = lm(y3~x3)
summary(model_yx3)
```

```
Call:
lm(formula = y3 ~ x3)

Residuals:
    Min      1Q  Median      3Q     Max
-3.5499 -0.8563  0.0292  0.9968  2.9554

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  -1.0671     0.1368  -7.798 6.83e-12 ***
x3            0.3940     0.1175   3.353  0.00114 **
---
Signif. codes:  0 ’***’ 0.001 ’**’ 0.01 ’*’ 0.05 ’.’ 0.1 ’ ’ 1
```

```
Residual standard error: 1.368 on 98 degrees of freedom
Multiple R-squared:  0.1029,    Adjusted R-squared:  0.09374
F-statistic: 11.24 on 1 and 98 DF,  p-value: 0.001139
```

```
abline(model_yx3, lwd=2, col="red")
abline(-1, 0.5, lwd=2, col="blue")
legend(-1, legend = c("model_yx3", "true relationship"), col=c("red","blue"), lwd
```



As expected, the model performance decreased, since there is more noise.

- *j):*

```
confint(model_yx)
```

```
                2.5 %     97.5 %
(Intercept) -1.1150804 -0.9226122
x            0.3925794  0.6063602
```

```
confint(model_yx2)
```

```
                2.5 %     97.5 %
(Intercept) -1.033141 -0.9451916
x2           0.481037  0.5664653
```

```
confint(model_yx3)
```

```
                2.5 %     97.5 %
(Intercept) -1.3386719 -0.7955438
x3           0.1607854  0.6272271
```

Confidence interval values are the smallest for the model with the lowest variance, largest for the highest variance model and in the middle of these two for the original model.

Exercises required for MSSC PhD students

1. **Simulation of bias-variance tradeoff.** Let $f(x) = x^2$ be the true regression function. Simulate the data using $y_i = f(x_i) + \epsilon_i, i = 1, \ldots, 100$, where $x_i = 0.01i$ and $\epsilon_i \overset{iid}{\sim} N(0, 0.3^2)$.

(a) Generate 250 training data sets.

Solution:

```
i = seq(1,100)
x = 0.01*i

training_data_sets <- replicate(
  250, data.frame(x = x, y = x2 + rnorm(100,0,0.3)),
                                  simplify = FALSE)
```

(b) For each data set, fit the following three models:

- Model 1: $y = \beta_0 + \beta_1 x + \epsilon$
- Model 2: $y = \beta_0 + \beta_1 x + \beta_2 x^2 + \epsilon$
- Model 3: $y = \beta_0 + \beta_1 x + \beta_2 x^2 + \cdots + \beta_9 x^9 + \epsilon$

Solution:

```
# First idea, problem: its of size 750
#models <- c()
#for( i in (1:250))
#{
#  model1 <- lm(y ~ x, data = training_data_sets[[i]])
#  model2 <- lm(y ~ x + I(x2), training_data_sets[[i]])
#  model3 <- lm(y ~ poly(x, 9), training_data_sets[[i]])
```

```
#  tobind <- list(model_1 = model1, model_2 = model2, model_3 = model3)
#  models <- c(models, tobind)
#}

# Better idea
models <- lapply(training_data_sets,
                 function(df)
                   {
                    model1 <- lm(y ~ x, data = df)
                    model2 <- lm(y ~ x + I(x2), data = df)
                    model3 <- lm(y ~ poly(x, 9), data = df)
                    list(model_1 = model1
                         , model_2 = model2, model_3 = model3)
                   }
                  )
```

(c) Calculate empirical MSE of $\hat{f}$, bias of $\hat{f}$ and variance of $\hat{f}$. Then show that

$$\text{MSE}_{\hat{f}} = \text{Bias}^2(\hat{f}) + \text{Var}(\hat{f}).$$

Solution:

```
f = function(x) {
  x  2
}
x0 = 0.9
y0 = f(x0)
predictions = matrix(0, nrow = 250, ncol = 3)
for (i in 1:250) {

  predictions[i, ] = c(
    predict(models[[i]]$model_1, newdata = data.frame(x = x0)),
    predict(models[[i]]$model_2, newdata = data.frame(x = x0)),
    predict(models[[i]]$model_3, newdata = data.frame(x = x0))
  )
}

get_bias = function(estimate, truth) {
  mean(estimate) - truth
}

get_mse = function(estimate, truth) {
  mean((estimate - truth)  2)
}
```

```r
get_var <- function(x) {
  n <- length(x)
  mean_x <- mean(x)
  sum_squared_differences <- sum((x - mean_x)2)
  sum_squared_differences / (n - 1)
}

bias = apply(predictions, 2, get_bias, f(x0))
variance = apply(predictions, 2, get_var)
mse = apply(predictions, 2, get_mse, y0)
cat("Model 1:", mse[1] ,"=", bias[1]2 + variance[1], "\n")
```

```
Model 1: 0.007144821 = 0.007153338
```

```r
cat("Model 2:", mse[2] ,"=", bias[2]2 + variance[2], "\n")
```

```
Model 2: 0.002763617 = 0.002774711
```

```r
cat("Model 3:", mse[3] ,"=", bias[3]2 + variance[3], "\n")
```

```
Model 3: 0.007045148 = 0.007073347
```

The results are fairly close to each other, for them to be exactly equal, our simulations should go towards infinity, or use n instead of n-1 when calculating variance.

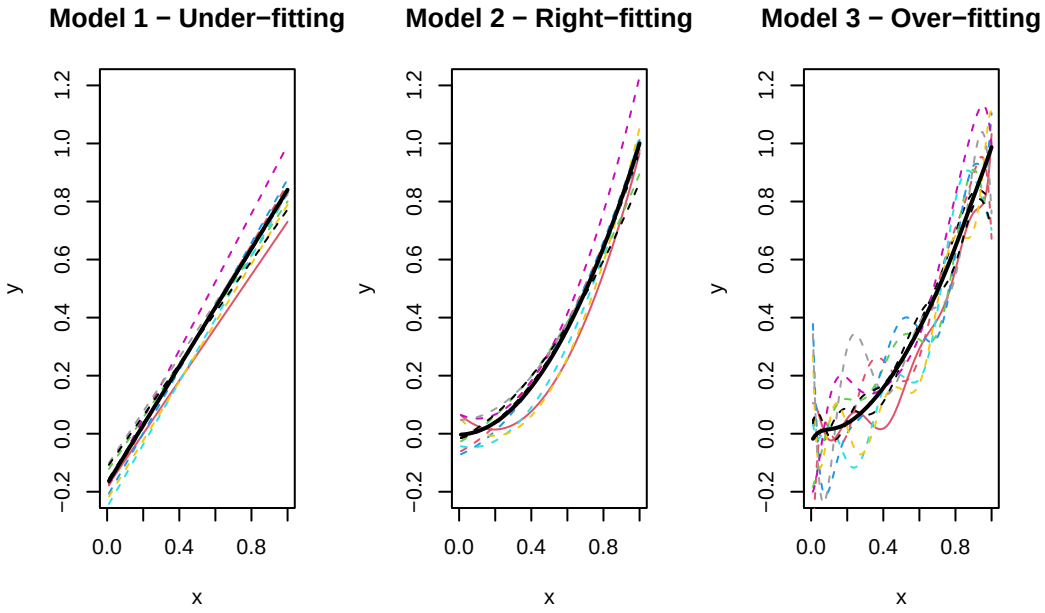
(d) For each model, plot first ten estimated f, $\hat{f}_1(x), \ldots, \hat{f}_{10}(x)$, and the average of $\hat{f}$, $\sum_{k=1}^{250} \hat{f}_k(x)$ in one figure, as Figure 1 below. What's your finding?

Solution:

```r
i = seq(1,100)
x = 0.01*i

sum_mod1 =  predict(models[[1]]$model_1, data=x)
for(i in 2:250)
  sum_mod1 <- sum_mod1 + predict(models[[i]]$model_1, data=x)
avg_mod1 <- sum_mod1/250

sum_mod2 =  predict(models[[1]]$model_2, data=x)
for(i in 2:250)
  sum_mod2 <- sum_mod2 + predict(models[[i]]$model_2, data=x)
```

```r
avg_mod2 <- sum_mod2/250

sum_mod3 =  predict(models[[1]]$model_3, data=x)
for(i in 2:250)
  sum_mod3 <- sum_mod3 + predict(models[[i]]$model_3, data=x)
avg_mod3 <- sum_mod3/250


par(mfrow = c(1, 3))
plot(x, predict(models[[1]]$model_1, data=x), type = "l", col = 10,
     ylim = c(-0.2, 1.2),xlim=c(0,1),
       main = paste("Model 1 - Under-fitting"), xlab = "x", ylab = "y")
lines(x, predict(models[[2]]$model_1, data=x), lty = 2, col = 1)
lines(x, predict(models[[3]]$model_1, data=x), lty = 2, col = 2)
lines(x, predict(models[[4]]$model_1, data=x), lty = 2, col = 3)
lines(x, predict(models[[5]]$model_1, data=x), lty = 2, col = 4)
lines(x, predict(models[[6]]$model_1, data=x), lty = 2, col = 5)
lines(x, predict(models[[7]]$model_1, data=x), lty = 2, col = 6)
lines(x, predict(models[[8]]$model_1, data=x), lty = 2, col = 7)
lines(x, predict(models[[9]]$model_1, data=x), lty = 2, col = 8)
lines(x, predict(models[[10]]$model_1, data=x), lty = 2, col = 9)
lines(x, avg_mod1, lty = 1, col = "black", lwd=2)

plot(x, predict(models[[1]]$model_2, data=x), type = "l", col = 10,
     ylim = c(-0.2, 1.2), xlim=c(0,1),
       main = paste("Model 2 - Right-fitting"), xlab = "x", ylab = "y")
lines(x, predict(models[[2]]$model_2, data=x), lty = 2, col = 1)
lines(x, predict(models[[3]]$model_2, data=x), lty = 2, col = 2)
lines(x, predict(models[[4]]$model_2, data=x), lty = 2, col = 3)
lines(x, predict(models[[5]]$model_2, data=x), lty = 2, col = 4)
lines(x, predict(models[[6]]$model_2, data=x), lty = 2, col = 5)
lines(x, predict(models[[7]]$model_2, data=x), lty = 2, col = 6)
lines(x, predict(models[[8]]$model_2, data=x), lty = 2, col = 7)
lines(x, predict(models[[9]]$model_2, data=x), lty = 2, col = 8)
lines(x, predict(models[[10]]$model_2, data=x), lty = 2, col = 9)
lines(x, avg_mod2, lty = 1, col = "black", lwd=2)

plot(x, predict(models[[1]]$model_3, data=x), type = "l", col = 10,
     ylim = c(-0.2, 1.2), xlim=c(0,1),
       main = paste("Model 3 - Over-fitting"), xlab = "x", ylab = "y")
lines(x, predict(models[[2]]$model_3, data=x), lty = 2, col = 1)
lines(x, predict(models[[3]]$model_3, data=x), lty = 2, col = 2)
lines(x, predict(models[[4]]$model_3, data=x), lty = 2, col = 3)
```

```
lines(x, predict(models[[5]]$model_3, data=x), lty = 2, col = 4)
lines(x, predict(models[[6]]$model_3, data=x), lty = 2, col = 5)
lines(x, predict(models[[7]]$model_3, data=x), lty = 2, col = 6)
lines(x, predict(models[[8]]$model_3, data=x), lty = 2, col = 7)
lines(x, predict(models[[9]]$model_3, data=x), lty = 2, col = 8)
lines(x, predict(models[[10]]$model_3, data=x), lty = 2, col = 9)
lines(x, avg_mod3, lty = 1, col = "black", lwd=2)
```



It is possible to see that the graph using models 1 has very high bias, and small variance. While the the one using models 3 is the opposite, it has a lower bias, but on the other hand, large variance. Models 2 do a better job, since it is the correct fit for the model, it balances variance and bias very well.

(e) Generate one more data set and use it as the test data. Calculate the training MSE and test MSE for each model.

Solution:

```
test_data <- x2 + rnorm(100,0,0.3)

calculate_mse <- function(model, data) {
  y_pred <- predict(model, data)
  mean((data$y - y_pred)2)
}
```

```
results <- list()
for (i in 1:250) {
  fit <- models[[i]]
  result <- data.frame(train_mse = numeric(3), test_mse = numeric(3))
  for (j in 1:3) {
    model <- fit[[j]]
    result[j, "train_mse"] <-calculate_mse(model, data = data.frame(x=training_da
    result[j, "test_mse"] <- calculate_mse(model, data =
data.frame(x=training_data_sets[[i]]$x, y=test_data))
  }
  results[[i]] <- result
}
```

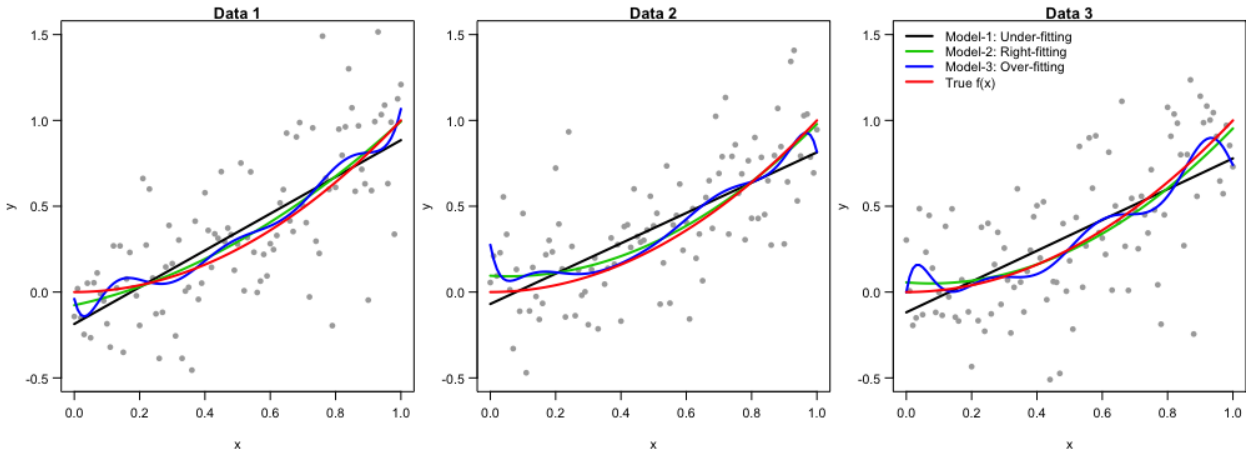


Figure 1: Trained f

2. **Gradient descent.** Write your own gradient descent algorithm for linear regression. Implement it and compare with the built-in function such as `optim()` to make sure that it converges correctly.

Solution:

```
gradient_Descent<-function(X, y, Beta, step_size){
  for(i in 1:5000)
  {
    Beta_prev <- Beta
    Beta<- Beta + step_size*(1/length(y))*(t(X)%*%(y-X%*%Beta))
    if(all(abs(Beta_prev-Beta) < 0.0001))
    {
      print("converged")
      return(Beta)
```

```
    }

  }
  return(Beta)
}


x <- matrix(rnorm(1000), ncol = 4)
y <-  10*x[,1]4+2*x[,2]3+4*x[,3]2+3*x[,4]

new_X<-cbind(rep(1, 100), x)
Beta<-rep(0,5)
step_size <- 1
Betas <- gradient_Descent(new_X, y, Beta, step_size)
```

```
[1] "converged"
```

```
#compare results
print(Betas)
```

```
           [,1]
[1,] 38.8948175
[2,] -0.7123358
[3,] 14.6454377
[4,] -7.8554077
[5,]  8.7330170
```

```
optimal <- lm(y~x)
print(coefficients(optimal))
```

```
(Intercept)          x1          x2          x3          x4
 38.8948344  -0.7123291  14.6454499  -7.8554152   8.7330254
```

As we can see, they are very close to each other, not exactly equal, since we are stopping the algorithm once it stops changing all results by 0.00001.

Optional Exercises

1. Show that $\text{MSE}_{y_0} = \text{E}\left[\left(y_0 - \hat{f}(x_0)\right)^2\right] = \text{MSE}_{\hat{f}} + \text{Var}(\epsilon)$

2. Prove the Gauss–Markov theorem: for linear regression the least squares estimate of a parameter $\mathbf{c}'\boldsymbol{\beta}$ has variance no bigger than that of any other linear unbiased estimate $\mathbf{c}'\boldsymbol{\beta}$.

3. Suppose we fit the model $\mathbf{y} = \mathbf{X}_1\boldsymbol{\beta}_1 + \boldsymbol{\epsilon}$ when the true model is actually $\mathbf{y} = \mathbf{X}_1\boldsymbol{\beta}_1 + \mathbf{X}_2\boldsymbol{\beta}_2 + \boldsymbol{\epsilon}$. For both models, assume $\mathrm{E}(\boldsymbol{\epsilon}) = \mathbf{0}$ and $\mathrm{Var}(\boldsymbol{\epsilon}) = \sigma^2\mathbf{I}$. Let $\mathbf{b}_1$ be the ordinary least-squares estimate of $\boldsymbol{\beta}_1$. Find the $\mathrm{E}(\mathbf{b}_1)$, $\mathrm{Var}(\mathbf{b}_1)$ and $\mathrm{MSE}(\mathbf{b}_1)$. Compare them with the ones from the true model. This is an underfitting example of linear regression.

4. Suppose we fit the model $\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \mathbf{Z}\boldsymbol{\gamma} + \boldsymbol{\epsilon}$ when the true model is actually $\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}$. For both models, assume $\mathrm{E}(\boldsymbol{\epsilon}) = \mathbf{0}$ and $\mathrm{Var}(\boldsymbol{\epsilon}) = \sigma^2\mathbf{I}$. Let $\mathbf{b}$ be the ordinary least-squares estimate of $\boldsymbol{\beta}$. Find the $\mathrm{E}(\mathbf{b})$, $\mathrm{Var}(\mathbf{b})$ and $\mathrm{MSE}(\mathbf{b})$. Compare them with the ones from the true model. This is an overfitting example of linear regression.