Problem 1

Bais-Variance Trade-off with Noise

Given $x = x_0$,

$$\begin{aligned}\mathbb{E}_{\mathcal{D}}[(g^{(\mathcal{D})}(x) - f(x) - \epsilon)^2] &= \mathbb{E}_{\mathcal{D}}[(g^{(\mathcal{D})} - \bar{g}(x) + \bar{g}(x) - f(x) - \epsilon)^2] \\ &= \mathbb{E}_{\mathcal{D}}[(g^{(\mathcal{D})}(x) - \bar{g}(x))^2] + (\bar{g}(x) - f(x))^2 + \\ &\quad 2\mathbb{E}_{\mathcal{D}}[g^{(\mathcal{D})}(x) - \bar{g}(x)](\bar{g}(x) - f(x)) + \epsilon^2 - \\ &\quad 2\mathbb{E}_{\mathcal{D}}[\epsilon(g^{(\mathcal{D})}(x) - \bar{g}(x))] - 2\mathbb{E}_{\mathcal{D}}[\epsilon](\bar{g}(x) - f(x))\end{aligned} \tag{1}$$

Note that by definition,

$$\begin{aligned}\mathbb{E}_{\mathcal{D}}[g^{(\mathcal{D})}(x) - \bar{g}(x)] &= 0 \\ \mathbb{E}[\epsilon] &= 0\end{aligned} \tag{2}$$

So all the cross-products equal zero. We hence get,

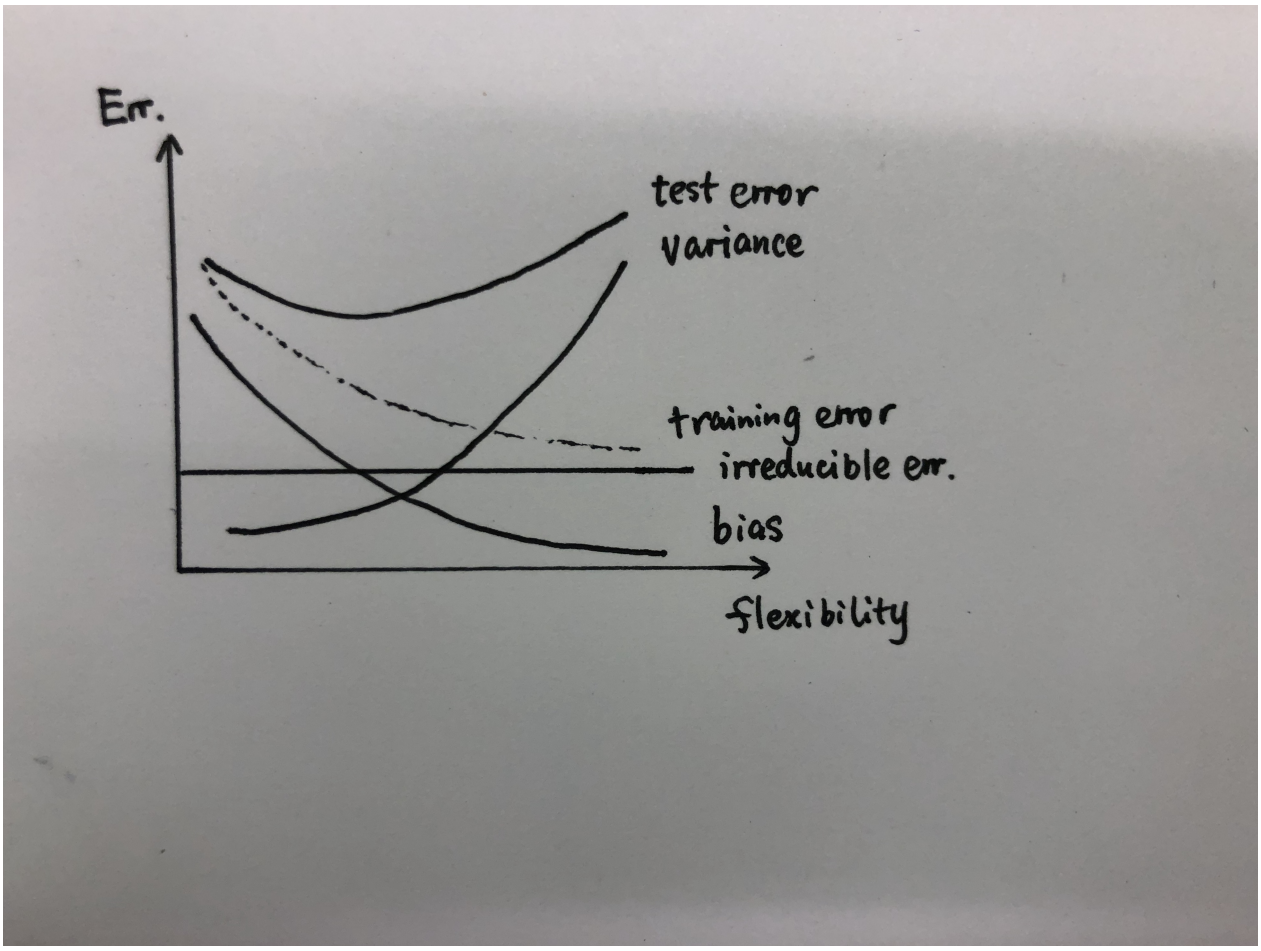
$$\begin{aligned}\mathbb{E}_{\mathcal{D}}[(g^{(\mathcal{D})}(x) - f(x) - \epsilon)^2] &= \mathbb{E}_{\mathcal{D}}[(g^{(\mathcal{D})}(x) - \bar{g}(x))^2] + [\bar{g}(x) - f(x)]^2 + \sigma^2 \\ &= bias^2 + var + \sigma^2\end{aligned} \tag{3}$$

Problem 2

Flexible or Inflexible

a. It's better to use flexible methods. Because flexible methods give us a higher chance of estimating f. Given the sample size is extremely large and p is small, we don't have to worry about degrees of freedom, nor overfitting.
b. It's better to use inflexible methods. Though flexible methods offers us less bias, it increases the variance at the same time. It is especially the case when the number of predictors p is large, and n is small. (Recall that in a linear model, this will affect the invertibility of X^TX) Using flexible methods will consume a lot of degrees of freedom and increase Overall MSE.
c. It's better to use flexible methods. Because inflexible methods are very unlikely to capture those non-linear relationship between predictors and response. Increasing the complexity of our model might gigantically decrease the bias.
d. It's better to use inflexible methods. When $Var(\epsilon)$ is really high, we're worried about overfitting. Though the flexible methods will lead us to a less in-sample bias, it's very likely that the out-of sample bias increases at the same time.

Problem 3



1. The (squared) bias shall be (generally) decreasing w.r.t. the amount of flexibility. Because when we're increasing the complexity of our model, we tend to capture more of the information in our training set. We'll have more opportunity to get unbiased estimation.
2. The (typical) variance shall be (generally) increasing w.r.t. the amount of flexibility. When the complexity of model increases, it poses a risk that the in-sample error no longer equals the out of sample error. Namely, we're facing a bias-variance trade-off.
3. The training error shall be generally decreasing, as an indicator of the model's ability to fit the sample.
4. The test error is a U-shape curve. It decreases at first as the bias of the estimation is lowered by the increase in flexibility. When the model became too complex, however, it overwhelms the benefit of increasing the flexibility and pushes up the test error.
5. The irreducible error shall remain generally a constant as it cannot be reduced by shifting the flexibility of the model.

Problem 4

The advantage of a flexible regression model is that it can capture more information in the training set, and thus offers people more opportunity to get lower bias. However, its disadvantage is that it poses some risk that out-of sample error might no longer equal in-sample data, namely, overfitting. By decreasing the degrees of freedom, the model estimation is tend to be suffering more variance.
Some of the classical scenarios where a flexible model might generally by preferred includes:
1. There are plentiful observations, the number of which is way larger than the number of predictors. $n >> k$ (Violating this will lead to a loss of degree of freedom and an increase in variance in the point-estimation of $\hat{\beta}$. Under some extreme circumastances, it'll even affect the invertibility of $X^T X$)
2. There are some solid theory behind the model, presuming a certain non-linear, more complicated relationship between our dependent variable and predictors.
3. The noise of the data is not too large, so that the information captured in the training set can be generalized to the original model.

Problem 5

The main difference between parametric models and non-parametric methods are that in a non-parametric estimation, you do not have to make assumption about the distribution of your data. You'll not impose a specification of your model before your estimation. It's not the case in parametric models, where you generally have fixed number of parameters to estimate. In terms of regressions, specifically, the main advantage of a parametric one is that it has more statistical power, where we can easily derive its confidence intervals and do statistical inference. It also saves us more computational power.
The advantage of non-parametric models are that they're more flexible, meaning that it's more possible for them to get a better (in-sample) fit of your data. They'll be stable under skewness and outliers, and you don't have to assume the distributions of them, offering them more power in small-sample estimation.

Problem 6

Experiment Design

Step 1. Randomly generate 20,000 test set in our interval, either with or without noise.
Step 2. In a 500-round loop, randomly choose 2 points every time as our training set. Fit a model with H_0: horizontal linear model and H_1: ordinary linear model with slope. Test it

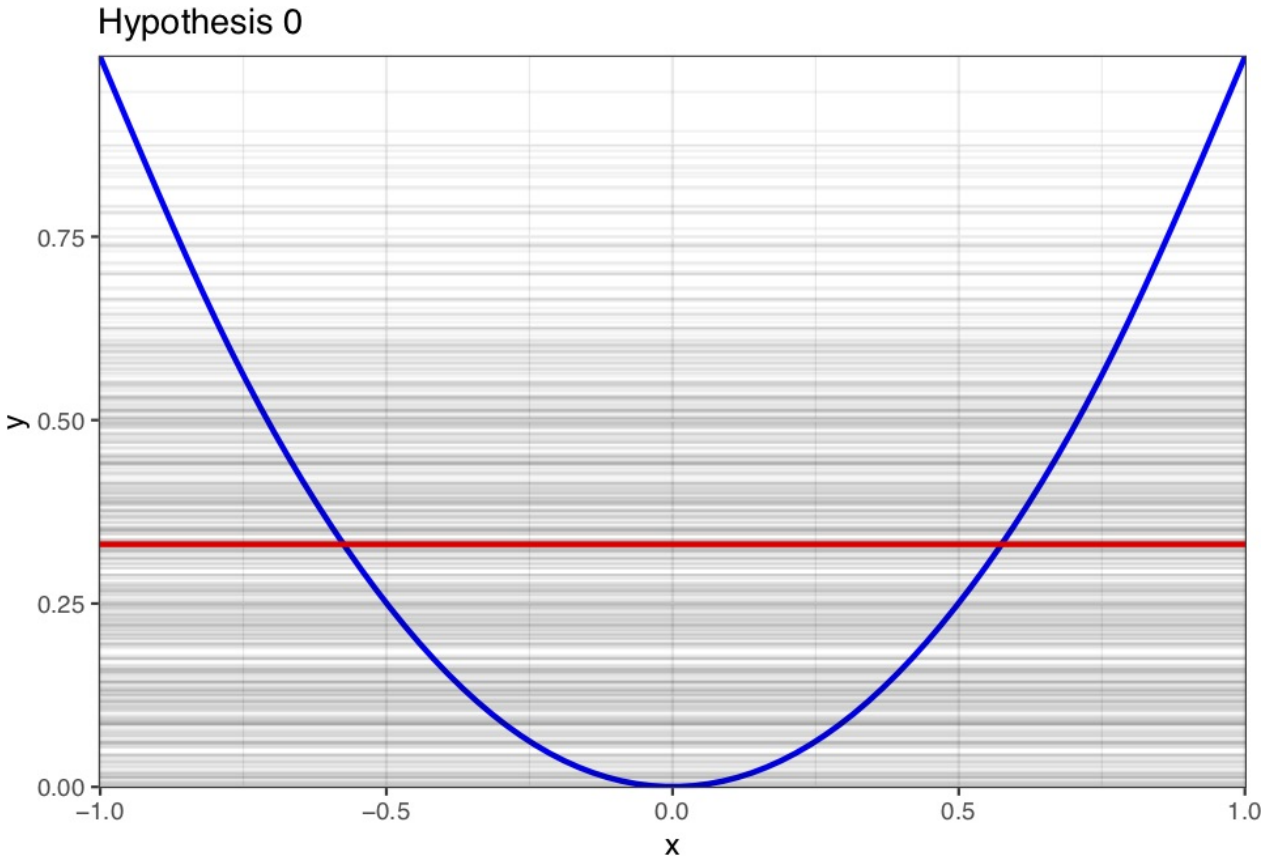
with our test set and store the coefficients.
Step 3. Calculate our average model for each scenario, compare the stability of our model with or without noise, plot the graph, and verify the bias-variance decomposition.

Code

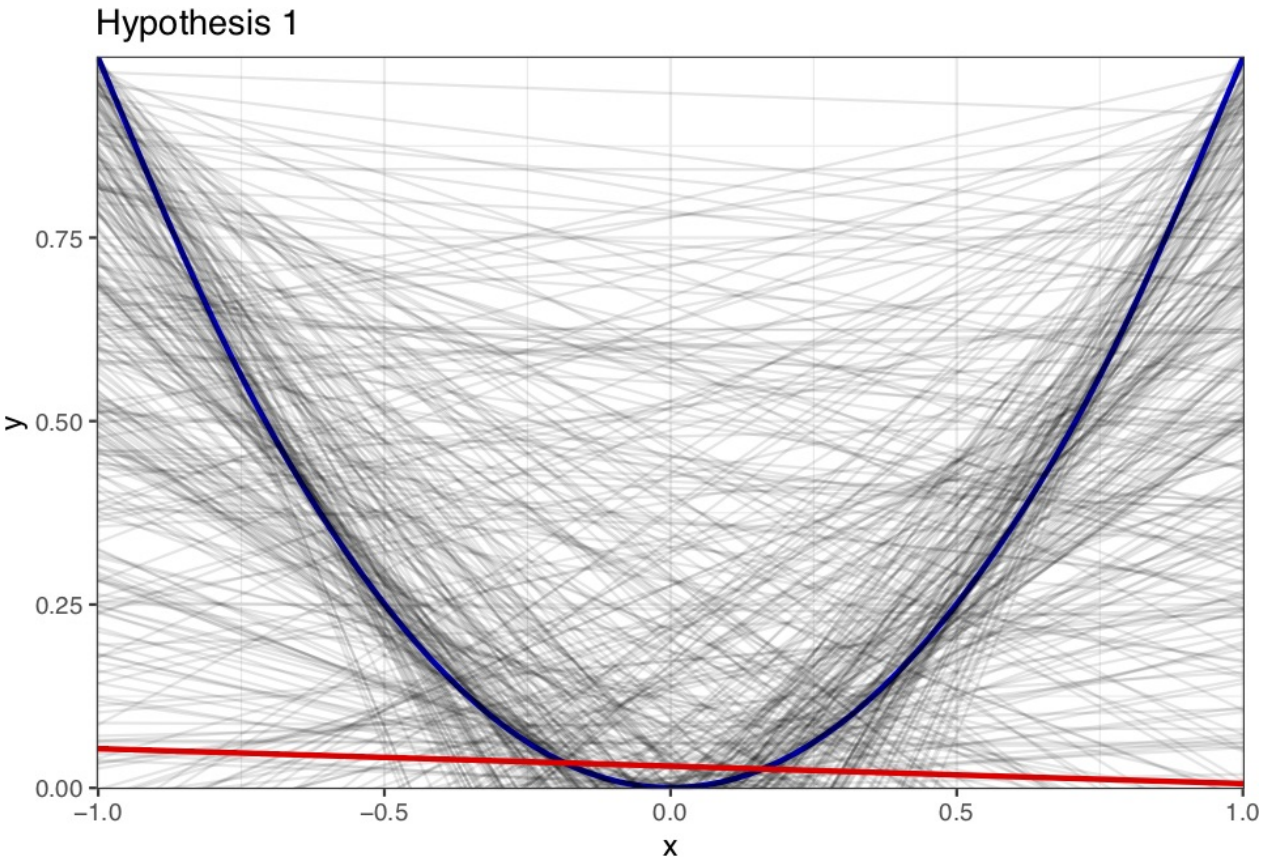
```
# Model H0: y = coef_0
# Model H1: y = coef_1a + coef_1b
n_out <- 20000
epsilon <- 0

# For each iteration, I'd like to record the following values
coef_0 <- vector()
coef_1a <- vector()
coef_1b <- vector()
MSE_0 <- rep(0, 500)
MSE_1 <- rep(0, 500)

# Here we randomly draw a test set
set.seed(12345)
x_out <- runif(n_out, min = -1, max = 1)
y_out <- x_out^2 + epsilon

# Fit the model with 2 training sets, and calculate the MSE of each model we fit
for (i in 1:500){
  x <- runif(n = 2, min = -1, max = 1)
  y <- x^2 + epsilon
  coef_0 <- c(coef_0, (y[1]+y[2])/2)
  coef_1a <- c(coef_1a, y[1] - x[1]*(y[2]-y[1])/(x[2]-x[1]))
  coef_1b <- c(coef_1b, (y[2]-y[1])/(x[2]-x[1]))
  for (j in 1:n_out){
    MSE_0[i] = MSE_0[i] + ((y_out[j] - coef_0[i])^2)/n_out
    MSE_1[i] = MSE_1[i] + ((y_out[j] - coef_1a[i] - coef_1b[i] * x_out[j])^2)/n_out
  }
}
```

```
# Here's the code for the H_0 plot
plot_data <- data.frame(x = x_out, y = y_out)
plot_data %>%
  ggplot() + theme_bw() +
  geom_smooth(aes(x = x, y = y), color = 'blue') +
```

```
  geom_hline(yintercept = coef_0, alpha = 0.05) +
  geom_hline(yintercept = mean(coef_0), color = 'red', size = 1) +
  scale_y_continuous(expand = c(0.001, 0.001)) +
  scale_x_continuous(expand = c(0.001, 0.001)) +
  labs(
    title = "Hypothesis 0"
  )
```




```
# Here's the code for the H_1 plot
plot_data %>%
  ggplot() + theme_bw() +
  geom_smooth(aes(x = x, y = y), color = 'blue') +
  geom_abline(intercept = coef_1a, slope = coef_1b, alpha = 0.1) +
  geom_abline(intercept = mean(coef_1a), slope = mean(coef_1b), color = 'red', size = 1)
  scale_y_continuous(expand = c(0.001, 0.001)) +
  scale_x_continuous(expand = c(0.001, 0.001)) +
  labs(
    title = "Hypothesis 1"
  )
```




```
# To verify the bias-variance trade-off rigorously, we examine the first model
OverallMSE_0 <- mean(MSE_0)
bias <- vector()
variance <- vector()
for (i in 1:20000){
  bias[i] <- (mean(coef_0) - y_out[i])^2
}
bias_0 <- mean(bias)
for (i in 1:500){
  variance[i] <- (mean(coef_0) - coef_0[i])^2
}
variance_0 <- mean(variance)
# Display it tidily
model_0 <- data.frame("OverallMSE" = OverallMSE_0, "Bias" = bias_0, "Var" = variance_0)
model_0
```

```
##   OverallMSE       Bias        Var
## 1  0.1330587 0.08903894 0.04401978
```

```
# If the decomposition is correct, it shall return 0
model_0$OverallMSE - model_0$Bias - model_0$Var
```

```
## [1] 6.938894e-18
```

```
# Similarly, we can do this to the second model
OverallMSE_1 <- mean(MSE_1)
bias <- vector()
variance <- rep(0, n_out)
for (i in 1:20000){
  bias[i] <- (mean(coef_1a) + mean(coef_1b)*x_out[i] - y_out[i])^2
}
bias_1 <- mean(bias)
for (i in 1:500){
  for (j in 1:n_out){
    variance[i] = variance[i] + ((mean(coef_1a) + mean(coef_1b)*x_out[j] - coef_1a[i] -
  }
}

variance_1 <- mean(variance)
# Display it tidily
model_1 <- data.frame("OverallMSE" = OverallMSE_1, "Bias" = bias_1, "Var" = variance_1)
model_1
```

```
##   OverallMSE      Bias       Var
## 1   0.486243 0.1802108 0.3060323
```

```
# If the decomposition is correct, it shall return 0
model_1$OverallMSE - model_1$Bias - model_1$Var
```

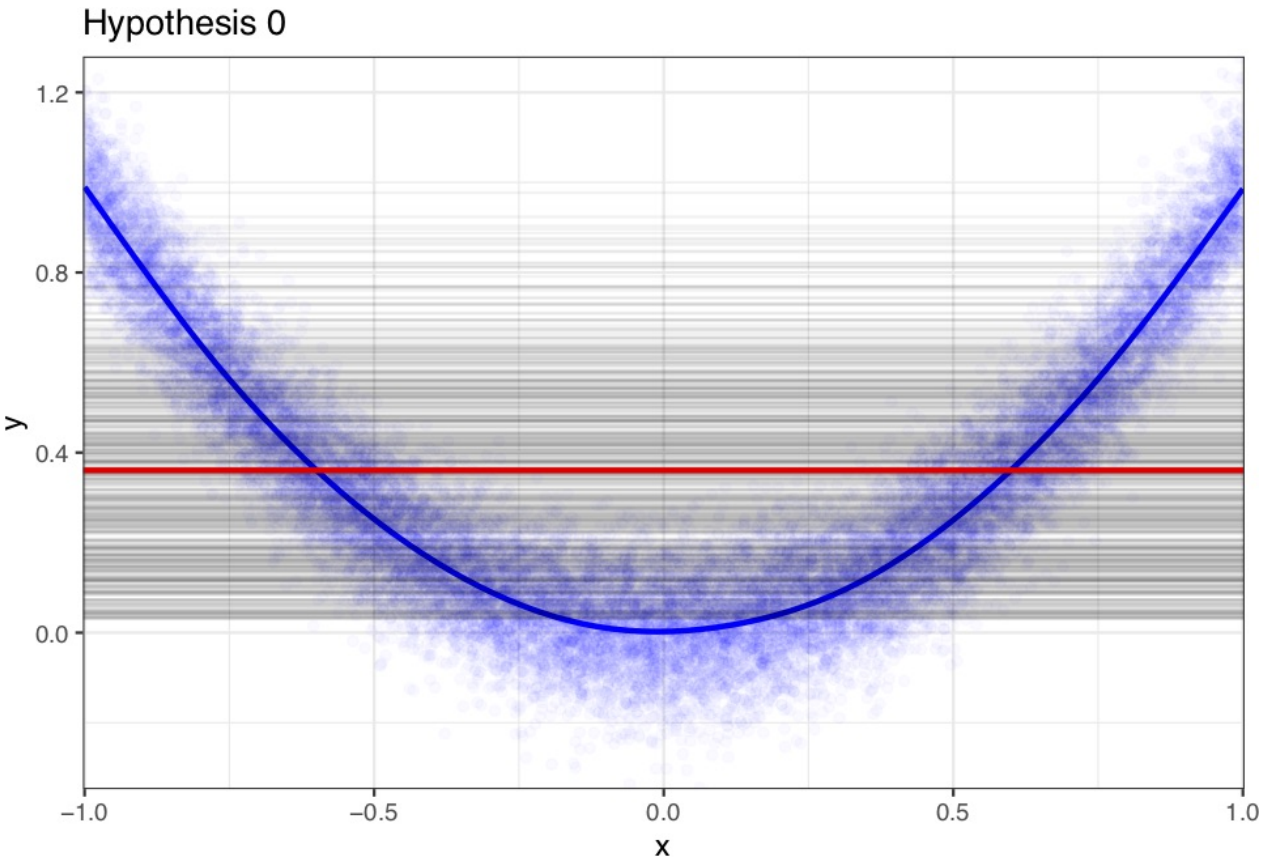
```
## [1] 0
```

Run another simulation with noise

```
# Model H0: y = coef_0
# Model H1: y = coef_1a + coef_1b
n_out <- 20000
epsilon <- rnorm(n_out, 0, 0.1)

# For each iteration, I'd like to record the following values
coef_0 <- vector()
coef_1a <- vector()
coef_1b <- vector()
MSE_0 <- rep(0, 500)
```

```
MSE_1 <- rep(0, 500)

# Here we randomly draw a test set
set.seed(12345)
x_out <- runif(n_out, min = -1, max = 1)
y_out <- x_out^2 + epsilon

# Fit the model with 2 training sets, and calculate the MSE of each model we fit
for (i in 1:500){
  x <- runif(n = 2, min = -1, max = 1)
  y <- x^2 + epsilon
  coef_0 <- c(coef_0, (y[1]+y[2])/2)
  coef_1a <- c(coef_1a, y[1] - x[1]*(y[2]-y[1])/(x[2]-x[1]))
  coef_1b <- c(coef_1b, (y[2]-y[1])/(x[2]-x[1]))
  for (j in 1:n_out){
    MSE_0[i] = MSE_0[i] + ((y_out[j] - coef_0[i])^2)/n_out
    MSE_1[i] = MSE_1[i] + ((y_out[j] - coef_1a[i] - coef_1b[i] * x_out[j])^2)/n_out
  }
}
```

```
# Here's the code for the H_0 plot
plot_data <- data.frame(x = x_out, y = y_out)
plot_data %>%
  ggplot() + theme_bw() +
  geom_point(aes(x = x, y = y), color = 'blue', alpha = 0.02) +
  geom_smooth(aes(x = x, y = y), color = 'blue') +
  geom_hline(yintercept = coef_0, alpha = 0.05) +
  geom_hline(yintercept = mean(coef_0), color = 'red', size = 1) +
  scale_y_continuous(expand = c(0.001, 0.001)) +
  scale_x_continuous(expand = c(0.001, 0.001)) +
  labs(
    title = "Hypothesis 0"
  )
```




```
# Here's the code for the H_1 plot
plot_data %>%
  ggplot() + theme_bw() +
  geom_smooth(aes(x = x, y = y), color = 'blue') +
  geom_point(aes(x = x, y = y), color = 'blue', alpha = 0.02) +
  geom_abline(intercept = coef_1a, slope = coef_1b, alpha = 0.05) +
  geom_abline(intercept = mean(coef_1a), slope = mean(coef_1b), color = 'red', size = 1)
  scale_y_continuous(expand = c(0.001, 0.001)) +
  scale_x_continuous(expand = c(0.001, 0.001)) +
  labs(
    title = "Hypothesis 1"
  )
```

```
# To verify the bias-variance trade-off rigorously, we examine the first model
OverallMSE_0 <- mean(MSE_0)
bias <- vector()
variance <- vector()
for (i in 1:20000){
  bias[i] <- (mean(coef_0) - x_out[i]^2)^2
```

```
}
bias_0 <- mean(bias)
for (i in 1:500){
  variance[i] <- (mean(coef_0) - coef_0[i])^2
}
variance_0 <- mean(variance)
# Display it tidily
model_0 <- data.frame("OverallMSE" = OverallMSE_0, "Bias" = bias_0, "Var" = variance_0)
model_0
```

```
##   OverallMSE       Bias        Var
## 1  0.1430415 0.08987592 0.04401978
```

```
# If the decomposition is correct, it shall return 0
model_0$OverallMSE - model_0$Bias - model_0$Var - 0.01
```

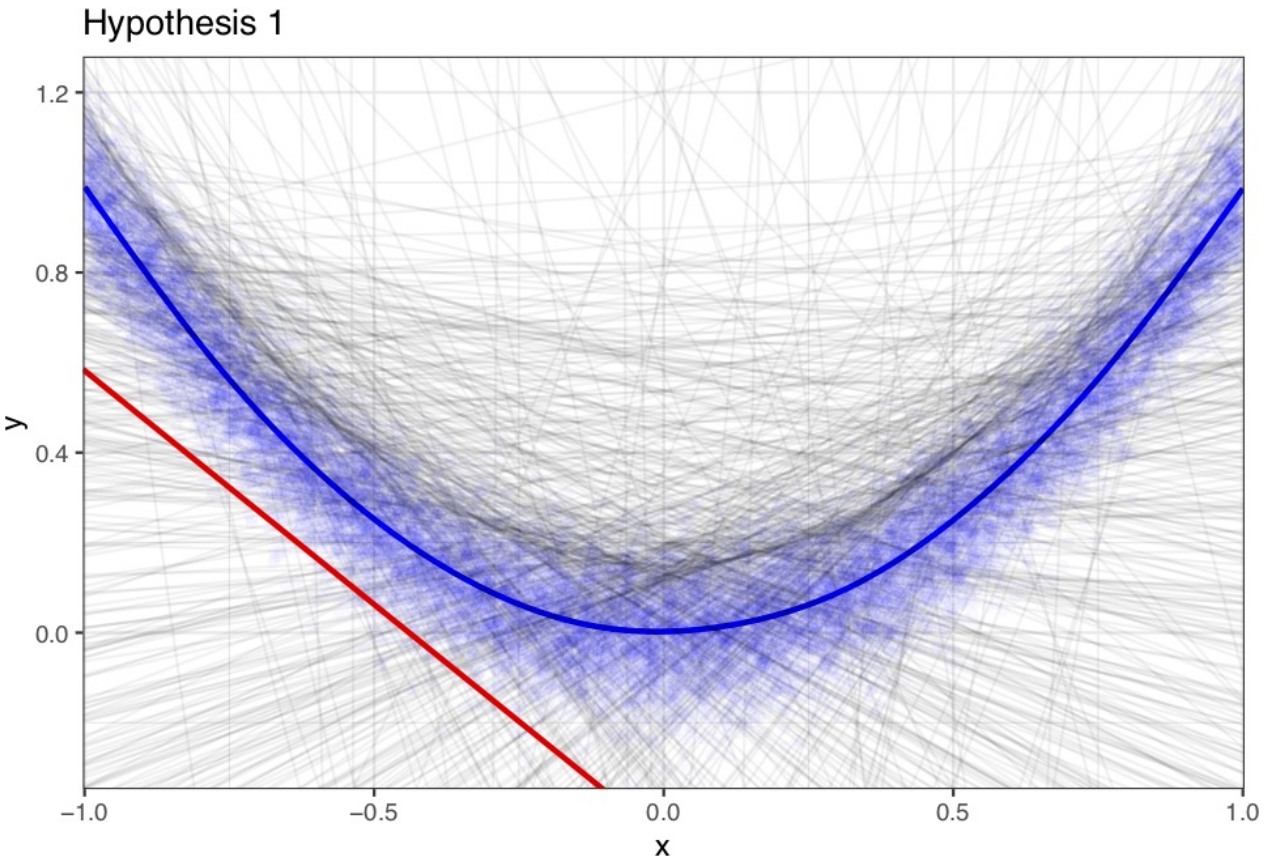
```
## [1] 0.000854249
```

```
OverallMSE_1 <- mean(MSE_1)
bias <- vector()
variance <- rep(0, n_out)
for (i in 1:20000){
  bias[i] <- (mean(coef_1a) + mean(coef_1b)*x_out[i] - x_out[i]^2)^2
}
bias_1 <- mean(bias)
for (i in 1:500){
  for (j in 1:n_out){
    variance[i] = variance[i] + ((mean(coef_1a) + mean(coef_1b)*x_out[j] - coef_1a[i] -
  }
}

variance_1 <- mean(variance)
# Display it tidily
model_1 <- data.frame("OverallMSE" = OverallMSE_1, "Bias" = bias_1, "Var" = variance_1)
model_1
```

```
##   OverallMSE     Bias      Var
## 1    298.159 1.061858 297.0879
```

```
# If the decomposition is correct, it shall return 0
model_1$OverallMSE - model_1$Bias - model_1$Var - 0.01
```

```
## [1] -0.000808448
```



Results

The results are displayed in the following table

Scenario	bias	variance	overall-MSE
$\sigma = 0$, Model H_0	0.089	0.044	0.133
$\sigma = 0$, Model H_1	0.180	0.306	0.486
$\sigma = 0.1$, Model H_0	0.090	0.044	0.143
$\sigma = 0.1$, Model H_1	1.062	297.09	298.16

Note that for the basic horizontal model, the performance is stable subject to noise. Its bias and variance remains roughly identical when noise is introduced. The more flexible model, however, is vulnerable to noises. Its variance bumps up tremendously in our last model, offering us a really unstable estimation.

Some explanation of the graphs

The blue curve is always our target function; the red line is our average estimation; the 10%-transparent blue dots around the blue curve are the randomly generated data (test set), and the grey lines are the fitted models from randomly generated training set.