R Lab: Bias-Variance Trade-Off

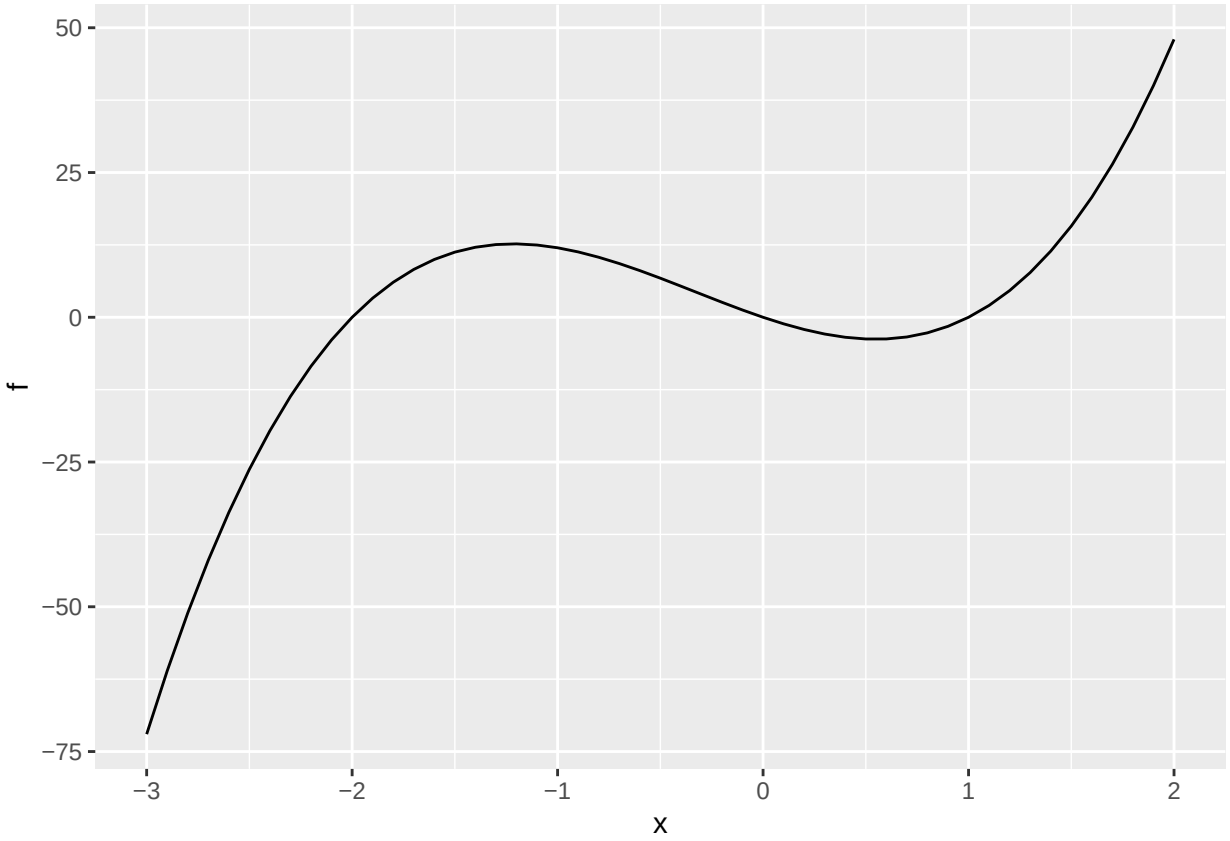
In this lab, we discuss one of the most important concepts of machine learning, which is the bias-variance trade-off. Although we use linear and polynomial regression as the machine learning models for this demonstration of the bias-variance trade-off, we can relate these methods to the dense feed-forward neural networks we have been using in the course.

As discussed earlier, linear regression can be written as a perceptron (i.e., a dense feed-forward neural network with no hidden layers) using a linear activation in the output node. Polynomial regression can be written as a dense feed-forward neural network with hidden layers using certain non-linear activation functions and a linear activation function in the output node. Increasing the degree of the polynomial regression model can be interpreted as increasing the number of hidden layers and/or units in the hidden layers. Thus, in the discussion below, we can think of linear regression as representing the simplest dense feed-forward neural network and higher-order polynomials as representing more complex dense feed-forward neural networks (with complexity increasing as the degree of the polynomial increases).

Bias-Variance Tradeoff

Assume that we have a regression problem where we are trying to fit a model to capture the relationship between one feature (the independent variable, x) and the response. Furthermore, assume that we happen to know the true relationship, given by the polynomial $f(x) = 6x^3 + 6x^2 - 12x$, shown below.

```
set.seed(123)
library(ggplot2)

true_relationship <- function(x) { return(6*x^3 + 6*x^2 - 12*x) }

x <- seq(-3, 2, by = 0.1)
f <- true_relationship(x)

ggplot() + geom_line(aes(x = x, y = f), color = "black")
```



The response consists of noisy values of the true relationship taken at the observation locations $x = -3, -2.9, \ldots, 2$. To simulate these noisy values, we add a normal random variable to each of the true relationship values.

```
observations <- f + rnorm(length(x), mean = 0, sd = 15)
```

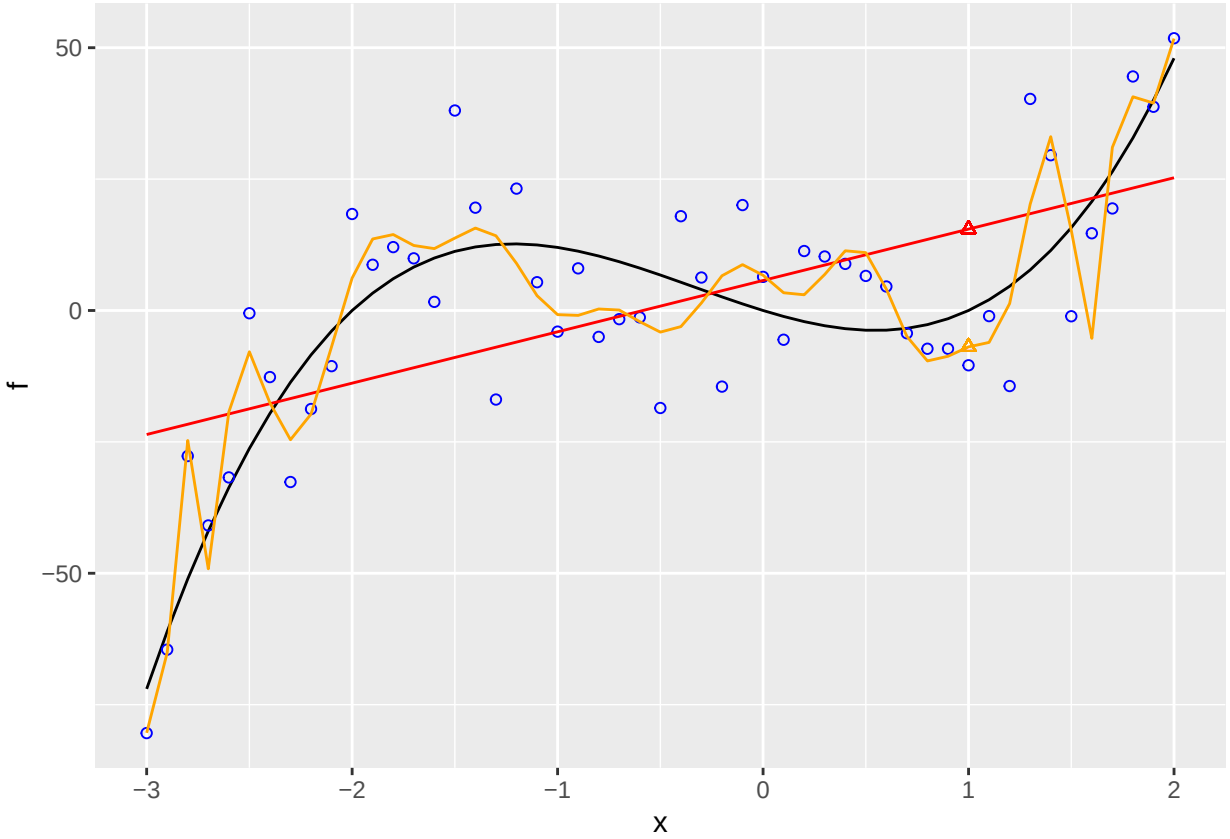
Now, let's try to capture the true relationship by fitting two polynomial regression models. The details behind polynomial regression are not important; it is enough to understand that a polynomial regression model seeks to find the best fitting polynomial, of a specified degree, to the data. We'll use a linear regression model (i.e., a 1-degree polynomial regression model) and a 25-degree polynomial regression model. We fit the polynomial regression models to the data using the 'lm' function.

```
model1 <- lm(observations ~ poly(x, 1))
predictions1 <- predict(model1, newdata = data.frame(x = x))

model25 <- lm(observations ~ poly(x, 25))
predictions25 <- predict(model25, newdata = data.frame(x = x))
```

The plot of the true relationship, the observations, the fitted linear regression model, and the fitted 25-degree polynomial regression model is shown below.

```
data <- data.frame(x = x,
                   f = f,
                   observations = observations,
                   lm = predictions1,
                   pm = predictions25)

ggplot(data, aes(x = x)) +
```

```
  geom_line(aes(y = f), color = "black") +
  geom_point(aes(y = observations), color = "blue", shape = 1) +
  geom_line(aes(y = lm), color = "red", linetype = "solid") +
  geom_line(aes(y = pm), color = "orange", linetype = "solid") +
  geom_point(aes(x = 1, y = data[x == 1, "lm"]), color = "red", shape=2) +
  geom_point(aes(x = 1, y = data[x == 1, "pm"]), color = "orange", shape=2)
```



From this plot, we can clearly see that neither regression model does an adequate job at capturing the true relationship. The linear regression model makes the assumption that the true relationship is linear, so it is not flexible enough and results in a model that cannot capture the structure of the relationship. Conversely, the 25-degree polynomial regression model makes the assumption that the true relationship is highly nonlinear, so it is too flexible and results in a model that is simply trying to match the observations instead of learning the structure of the relationship

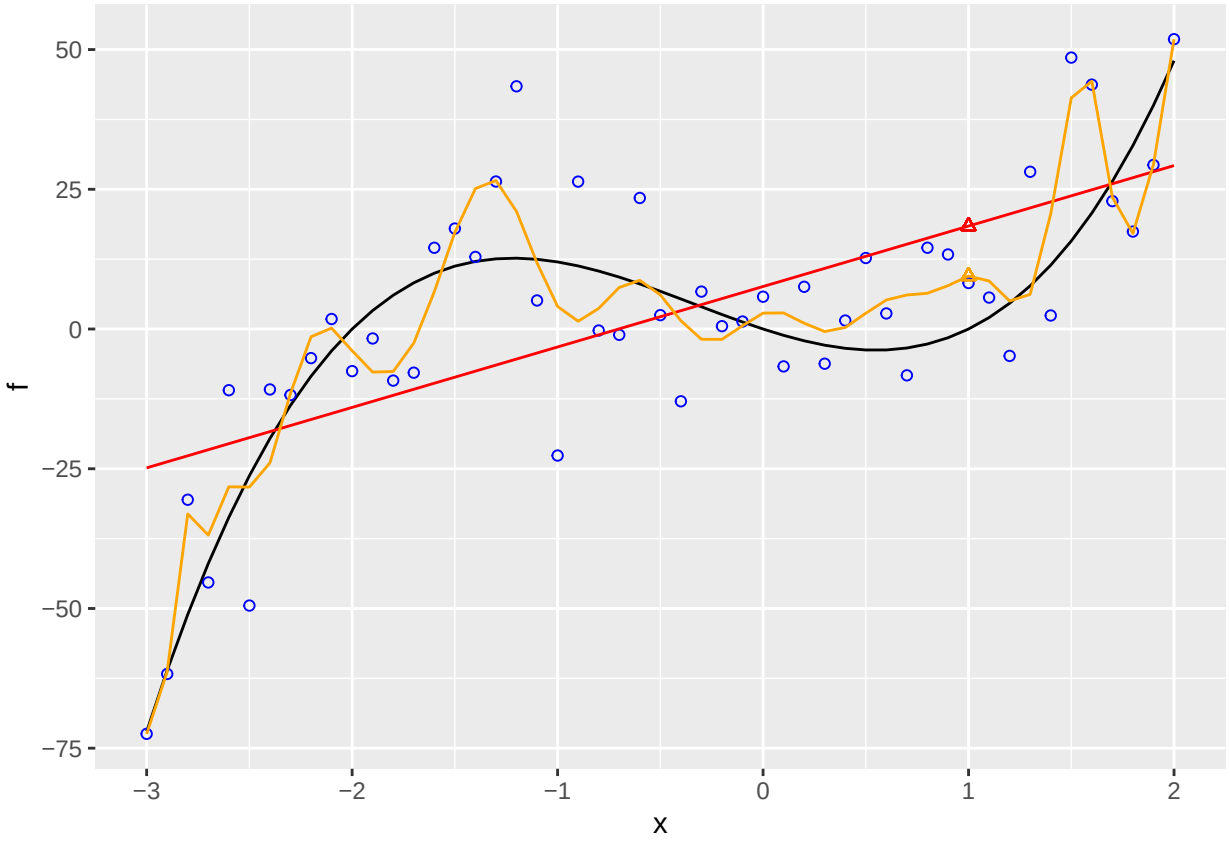
The plot also shows the predictions that each model makes at $x = 1$, denoted by the triangle markers. Note the positions of these predictions, since we will now use a new set of noisy values taken at the same observation locations $x = -3, -2.9, \ldots, 2$ to build the polynomial regression models.

```
observations_new <- f + rnorm(length(x), mean = 0, sd = 15)

model1 <- lm(observations_new ~ poly(x, 1))
predictions1 <- predict(model1, newdata = data.frame(x = x))

model25 <- lm(observations_new ~ poly(x, 25))
predictions25 <- predict(model25, newdata = data.frame(x = x))
```

The behavior of the new polynomial regression models, shown below, is the same as when we used the

previous data. Namely, the linear regression model is not flexible enough and the 25-degree polynomial model is too flexible.

```
data <- data.frame(x = x,
                   f = f,
                   observations = observations_new,
                   lm = predictions1,
                   pm = predictions25)

ggplot(data, aes(x = x)) +
  geom_line(aes(y = f), color = "black") +
  geom_point(aes(y = observations), color = "blue", shape = 1) +
  geom_line(aes(y = lm), color = "red", linetype = "solid") +
  geom_line(aes(y = pm), color = "orange", linetype = "solid") +
  geom_point(aes(x = 1, y = data[x == 1, "lm"]), color = "red", shape = 2) +
  geom_point(aes(x = 1, y = data[x == 1, "pm"]), color = "orange", shape = 2)
```



If we compare the predictions at $x = 1$ from both sets of polynomial regression models, we make the following observations:

1) The predictions from the two linear regression models are similar. Furthermore, the center of both predictions is not close to the value of the true relationship at $x = 1$.
2) The predictions from the two 25-degree polynomial regression models are very different. However, the center of both predictions is close to the value of the true relationship at $x = 1$. In other words, the predictions seem to be centered around the value of the true relationship.

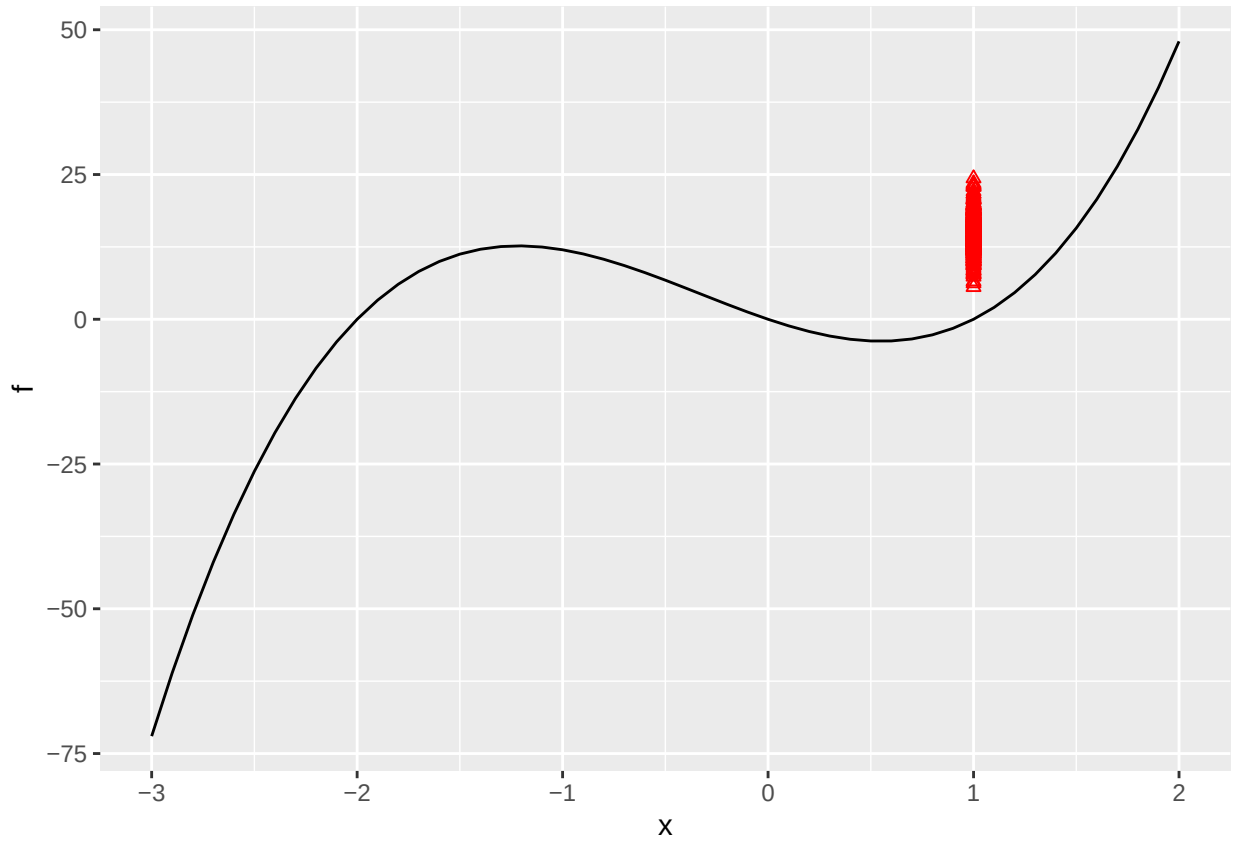
The behavior of these predictions is due to the bias-variance trade-off: the predictions from the linear regression model have higher bias but lower variance, while the predictions from the 25-degree polynomial

regression model have lower bias but higher variance. Bias is the difference between the average predictions of the model and the value of the true relationship. Here, the linear regression model has high bias, since it makes the assumption that the true relationship is linear and leads to a model that is too simple to capture the relationship. Variance is the amount of variability in the model predictions. Here, the 25-degree polynomial regression model has high variance since it tries to match the observations too closely, so even a slight change in the observations will lead to predictions that are very different.

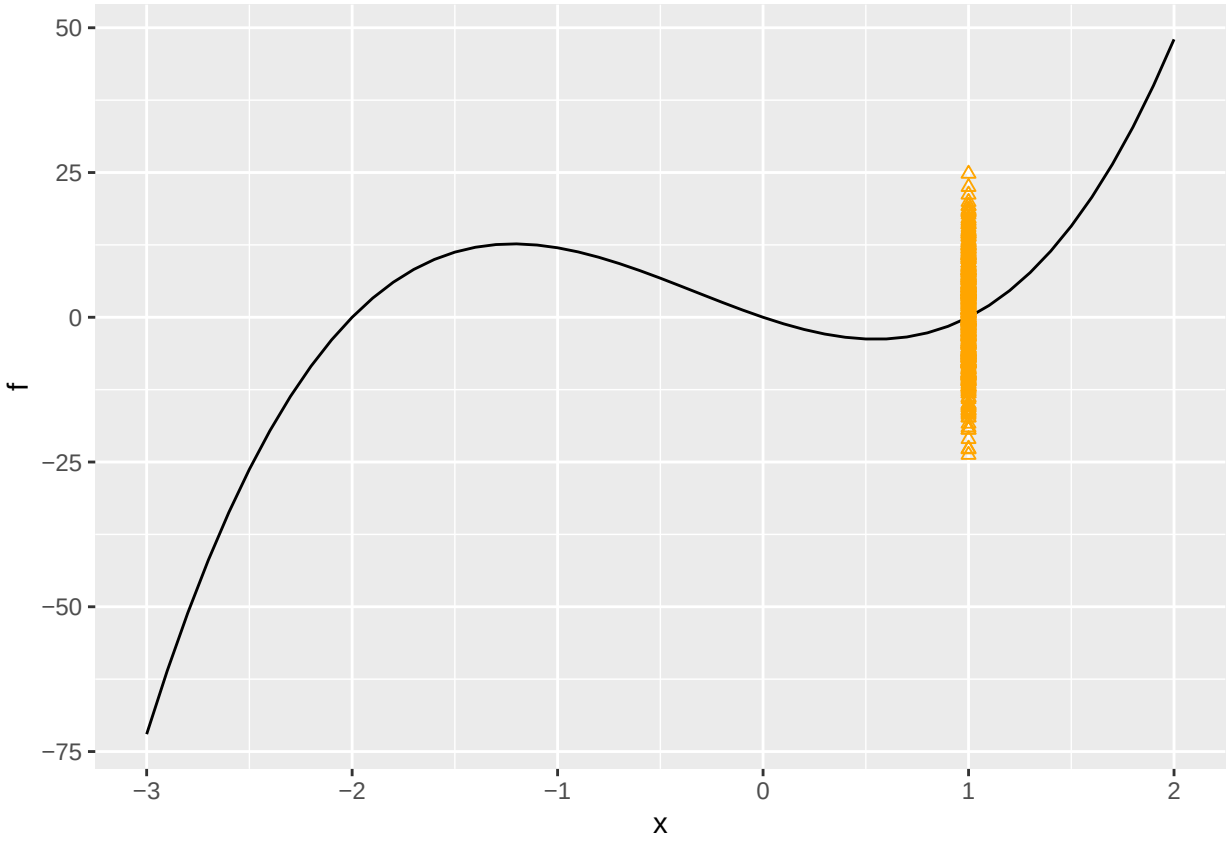
In other words, the linear regression model pays very little attention to the observations, whereas the 25-degree polynomial regression model pays too close attention to the observations. A slight change in the observations will lead to a much larger change in the predictions of the 25-degree polynomial regression model than in the linear regression model.

To get more intuition behind the bias-variance trade-off, let's perform the same experiment 500 times to get 500 predictions from each polynomial regression model at $x = 1$.

```
results1 <- data.frame(x = 1, f_pred = 0)
for (i in 1:500) {
  x <- seq(-3, 2, by = 0.1)
  f <- true_relationship(x)

  temp_observations <- f + rnorm(length(x), mean=0, sd=15)

  model1 <- lm(temp_observations ~ poly(x, 1))
  results1[i, 1] <- 1
  results1[i, 2] <- predict(model1, newdata = data.frame(x=1))
}

ggplot() +
  geom_line(data = data, aes(x = x, y = f), color = "black") +
  geom_point(data = results1, aes(x = x, y = f_pred), color="red", shape=2)
```



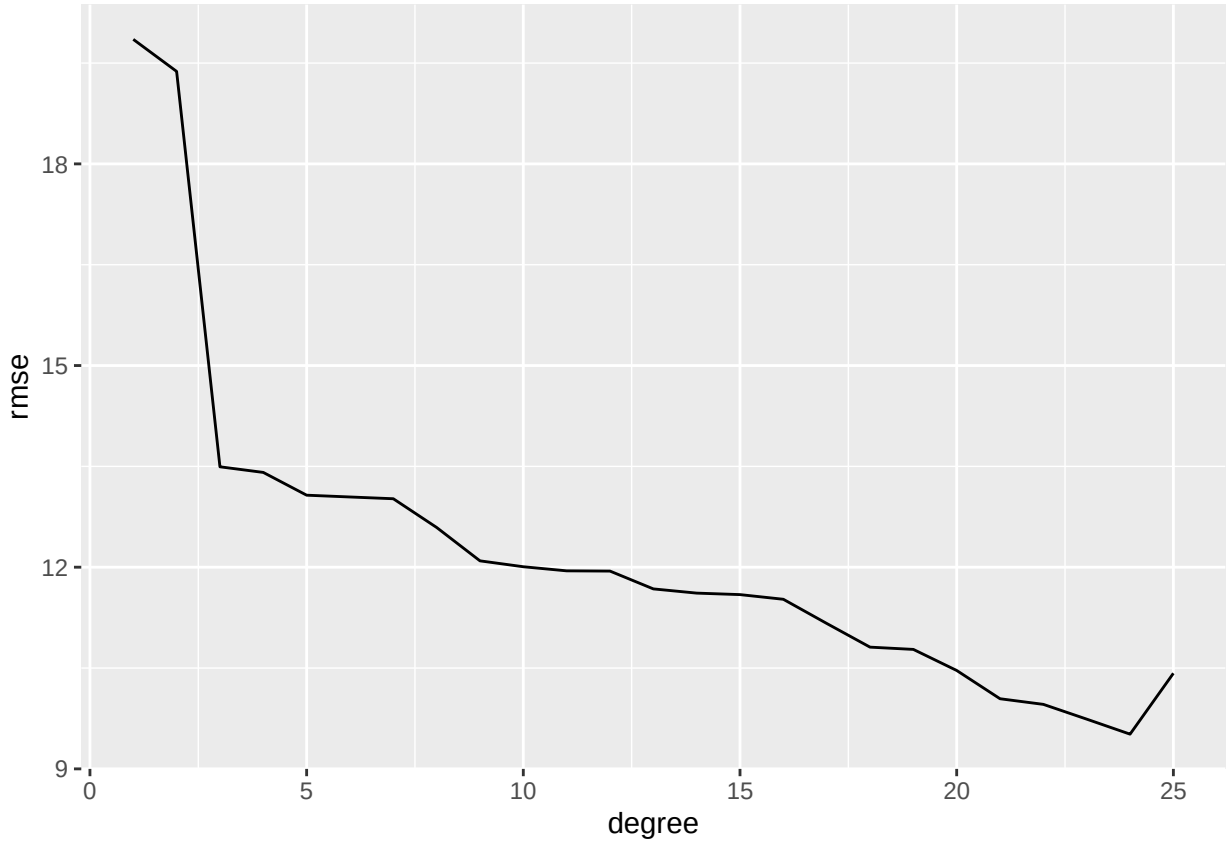

```
results20 <- data.frame(x = 1, f_pred = 0)
for (i in 1:500) {
  x <- seq(-3, 2, by = 0.1)
  f <- true_relationship(x)

  temp_observations <- f + rnorm(length(x), mean=0, sd=15)

  model20 <- lm(temp_observations ~ poly(x, 20))
  results20[i, 1] <- 1
  results20[i, 2] <- predict(model20, newdata = data.frame(x=1))
}

ggplot() +
  geom_line(data = data, aes(x = x, y = f), color = "black") +
  geom_point(data = results20, aes(x = x, y = f_pred), color="orange", shape=2)
```



Now, we can clearly see the bias-variance trade-off: the predictions from the linear regression model (in red) are less dispersed, but are not centered around the value of the true relationship at $x = 1$, whereas the predictions from the 25-degree polynomial regression model (in orange) are more dispersed, but are centered around the value of the true relationship at $x = 1$.

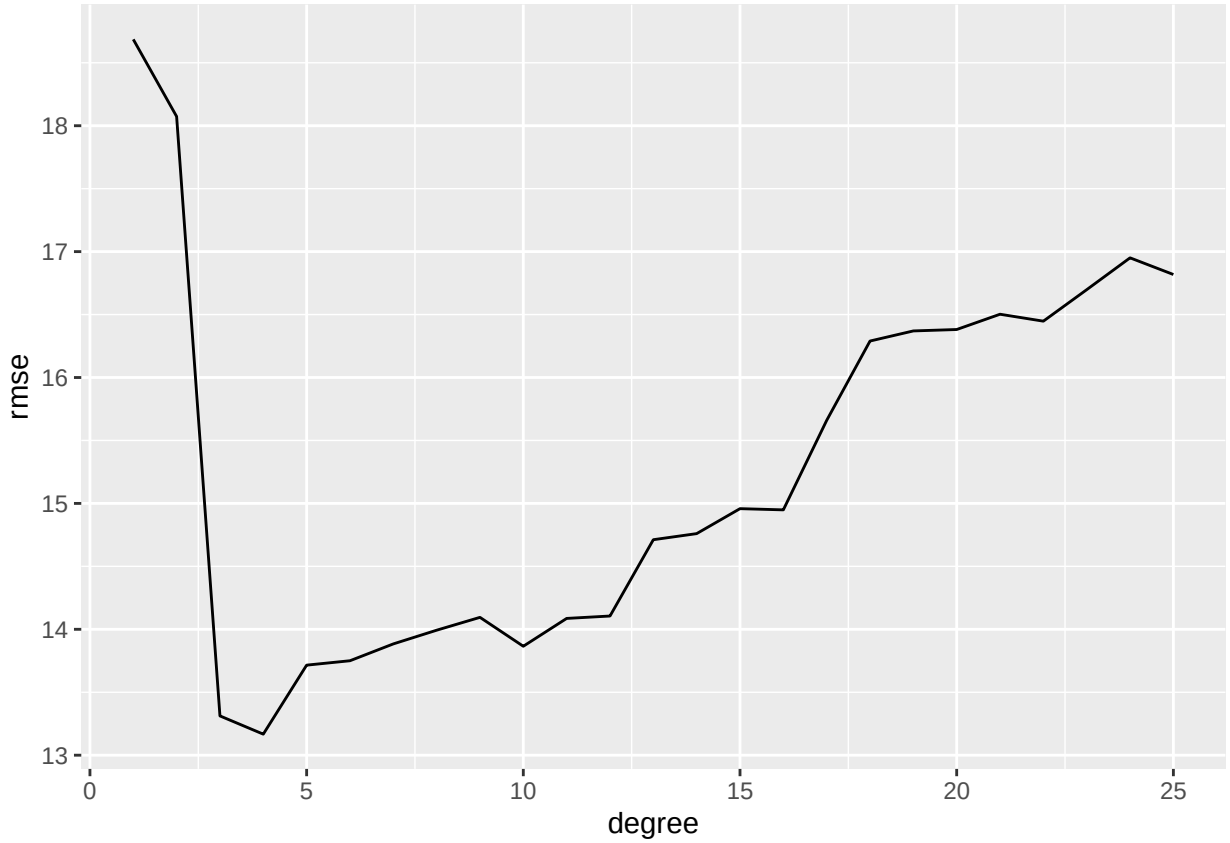
To obtain the best polynomial regression model, we need to find a degree of polynomial between the two that manages the bias-variance trade-off. To do this, let's fit polynomials with degrees ranging from 1 to 25 using the original data (in the dataframe called 'observations'). We'll also use the original data to evaluate the polynomials. The evaluation metric we use is called root mean squared error (RMSE), which measures how close the model predictions are to the observed values. We want to minimize RMSE. The plot of RMSE for each of the polynomial regression models is shown below.

```
models <- vector("list", 25)
for (degree in 1:25) {
  model <- lm(observations ~ poly(x, degree))
  models[[degree]] <- model
}

results <- data.frame(degree = 1:25, rmse = 0)
for (degree in 1:25) {
  predictions <- predict(models[[degree]], newdata = data.frame(x=x))
  results[results$degree==degree, "rmse"] <-
    sqrt((1/length(predictions))*sum((predictions-observations)^2))
}

ggplot() +
  geom_line(data = results, aes(x = degree, y = rmse), color = "black")
```



From this plot, the best models appear to be the polynomials of highest degree, since they result in a lower RMSE. However, we need to remember that the higher-degree polynomials are simply trying to match the observations used to fit them (see the discussion above). Instead of fitting and evaluating the polynomial regression models on the same data, let's fit them using the original observations (in the dataframe 'observations') and evaluate them using the new observations (in the dataframe 'observations_new'). The plot of RMSE for each of the polynomial regression models, now using the new observations to calculate RMSE, is shown below.

```
results <- data.frame(degree = 1:25, rmse = 0)
for (degree in 1:25) {
  predictions <- predict(models[[degree]], newdata = data.frame(x=x))
  results[results$degree==degree, "rmse"] <-
    sqrt((1/length(predictions))*sum((predictions-observations_new)^2))
}

ggplot() +
  geom_line(data = results, aes(x = degree, y = rmse), color = "black")
```



Now, we see very different behavior.

Exercises

1) Why do the two RMSE plots show very different behavior? Use underfitting and overfitting in your answer, as well as the bias-variance tradeoff.

2) Using the last plot above, what degree of polynomial should we choose? After running all of the above code, substitute the degree you chose for the question mark in the code below and run it. Copy and paste the resulting plot.

```
model <- lm(observations ~ poly(x, ?))
predictions=predict(model, newdata = data.frame(x=x))
data = data.frame(x=x, f=f, predictions=predictions)
ggplot(data, aes(x=x)) +
  geom_line(aes(y = f), color = "black") +
  geom_line(aes(y = predictions), color = "red", linetype="solid")
```