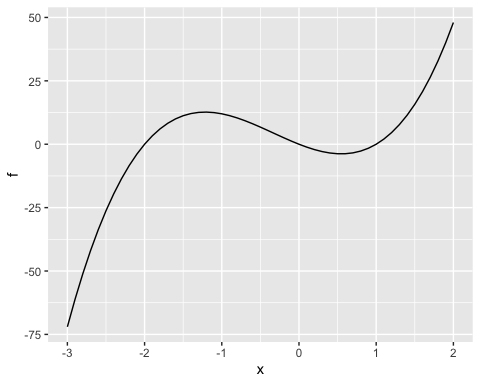
Lab 6

Adeline Casali

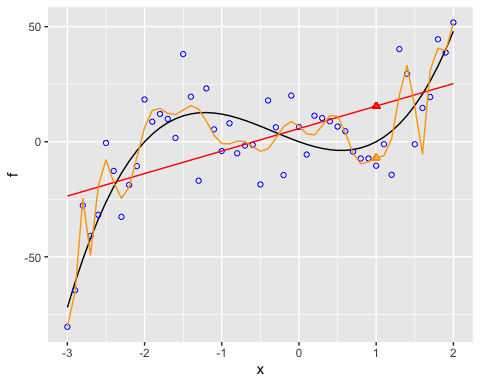
2024-02-23

Bias Variance Trade-Off

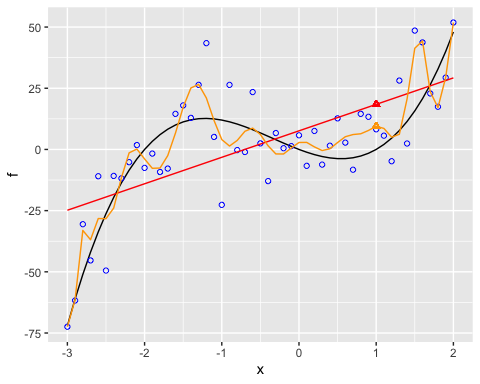
# Establishing and visualizing the true relationship  
set.seed(123)  
library(ggplot2)  
  
true\_relationship <- function(x) { return(6\*x^3 + 6\*x^2 - 12\*x) }  
  
x <- seq(-3, 2, by = 0.1)  
f <- true\_relationship(x)  
  
ggplot() + geom\_line(aes(x = x, y = f), color = "black")



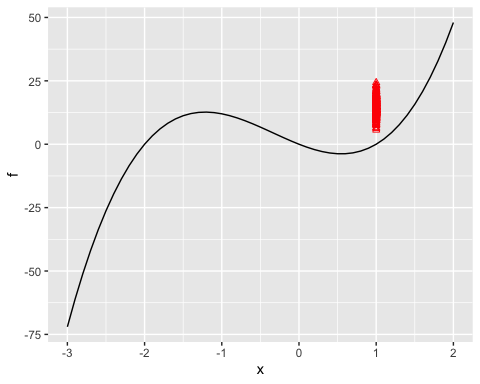
# Simulating noisy values  
observations <- f + rnorm(length(x), mean = 0, sd = 15)  
  
# Fitting two polynomial regression models  
model1 <- lm(observations ~ poly(x, 1))  
predictions1 <- predict(model1, newdata = data.frame(x = x))  
  
model25 <- lm(observations ~ poly(x, 25))  
predictions25 <- predict(model25, newdata = data.frame(x = x))  
  
data <- data.frame(x = x,   
 f = f,   
 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)



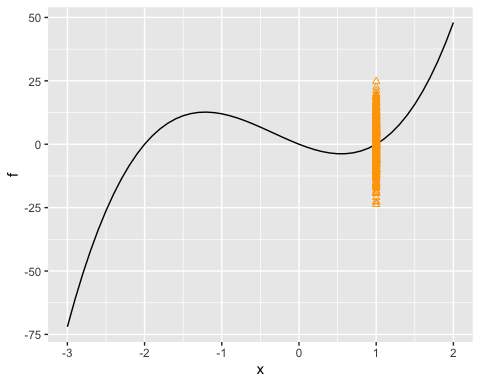
# Building two new models  
observations\_new <- f + rnorm(length(x), mean = 0, sd = 15)  
  
model1 <- lm(observations\_new ~ poly(x, 1))  
predictions1 <- predict(model1, newdata = data.frame(x = x))  
  
model25 <- lm(observations\_new ~ poly(x, 25))  
predictions25 <- predict(model25, newdata = data.frame(x = x))  
  
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)



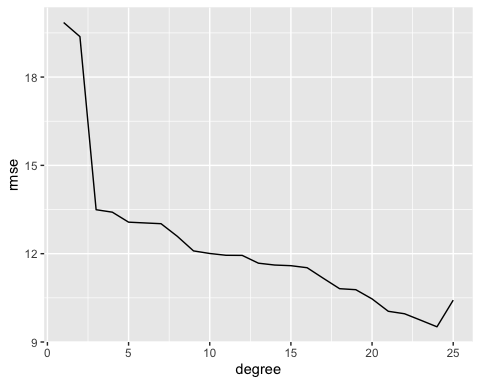
# Repeating 500 times (lm)  
results1 <- data.frame(x = 1, f\_pred = 0)  
for (i in 1:500) {  
 x <- seq(-3, 2, by = 0.1)  
 f <- true\_relationship(x)  
   
 temp\_observations <- f + rnorm(length(x), mean = 0, sd = 15)  
   
 model1 <- lm(temp\_observations ~ poly(x, 1))  
 results1[i, 1] <- 1  
 results1[i, 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)



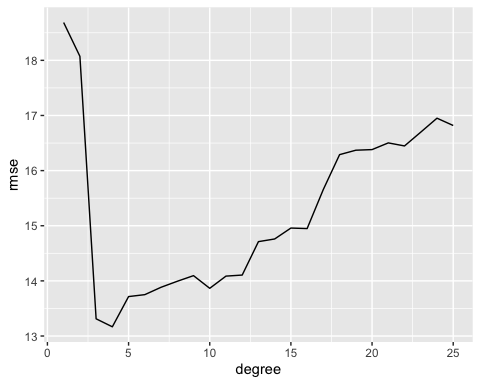
# Repeating 500 times (pm)  
results20 <- data.frame(x = 1, f\_pred = 0)  
for (i in 1:500) {  
 x <- seq(-3, 2, by = 0.1)  
 f <- true\_relationship(x)  
   
 temp\_observations <- f + rnorm(length(x), mean = 0, sd = 15)  
   
 model20 <- lm(temp\_observations ~ poly(x, 20))  
 results20[i, 1] <- 1  
 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)



# Creating and evaluating a range of models  
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")



# Same thing, but evaluated with observations\_new  
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")



1. In the first plot, the steady decrease in RMSE as the polynomial degree increases suggests overfitting, where the model captures excess noise in the data, leading to high variance. For this plot, the most flexible models with the highest polynomial degree appear to be the best models when they actually are overfitting to match the observations used to fit them. On the other hand, the models in the second plot are evaluated on a new set of observations, which shows that the true balance between bias and variance lies at 4th degree polynomial model, with the lowest RMSE. The models with degrees lower than 4 are too simplistic to capture underlying patterns (underfitting), resulting in high bias, while the models with degrees higher than 4 are overfitting, leading to high variance. Overall, this difference highlights the importance of finding the right level of model complexity through the bias-variance tradeoff, to achieve optimal generalization performance.
2. The model with the lowest RMSE was the 4th degree polynomial model. As seen from the plot below, it follows the true function extremely closely.

model <- lm(observations ~ poly(x, 4))  
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")

