Homework-4: Preston Robertson

All of these homework problems are from the *R for Data Science* book. The section numbers (e.g., “3.2.4 Exercises”) refer to sections in this book. Although the questions are based on those in the book, some questions ask for additional details or analysis.

When solving these problems, you are allowed to use any method from the book or class, even if that method wasn’t yet covered when the exercise was presented in the book.

Write answers that are as complete as possible. If a graph is helpful for formalizing the solution, provide the graph. If a table is helpful, provide a table. In the text part of the answer, outline the progression in your thinking as you perform the analysis.

Note that you should type your answers in RStudio, by typing into the file **Homework-4.sa.Rmd**.

## 23.2.1 Exercises

### (1) 23.2.1 Exercise 1

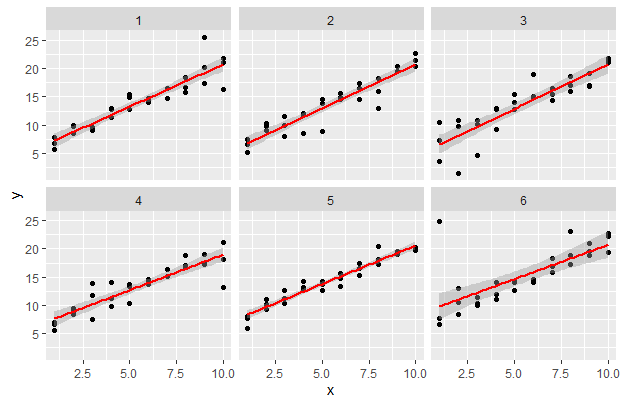
One downside of the linear model is that it is sensitive to unusual values because the distance incorporates a squared term. Fit a linear model to the simulated data below, and visualize the results. Rerun everything 6 times to generate different simulated datasets, and also examine the slope and intercept of the generated models, as well as the values. How much variability do you see? How do the data values affect the model?

# Changed to a function to make it iteratable, and added a variable for iterations.  
  
sim1a <- function(i){  
 tibble(  
 x = rep(1:10, each = 3),  
 y = x \* 1.5 + 6 + rt(length(x), df = 2),  
 .id = i  
 )  
}

*Hint*: To make it easy to do this 6 times, it makes sense to wrap everything into a function, and call that from within a for loop. The following code will help you accomplish this:

# Below we can see very high variability between each run in terms of correctness. In terms of the slope however, the linear models seem to portray the same slope. The data has a huge effect on the model, and that makes sense because the machine learning techniques are based on these models.  
  
sims <- map\_df(1:6, sim1a)  
  
ggplot(sims, aes(x = x, y = y)) +  
 geom\_point() +  
 geom\_smooth(method = "lm", colour = "red") +  
 facet\_wrap(~.id, ncol = 3)

## `geom\_smooth()` using formula 'y ~ x'



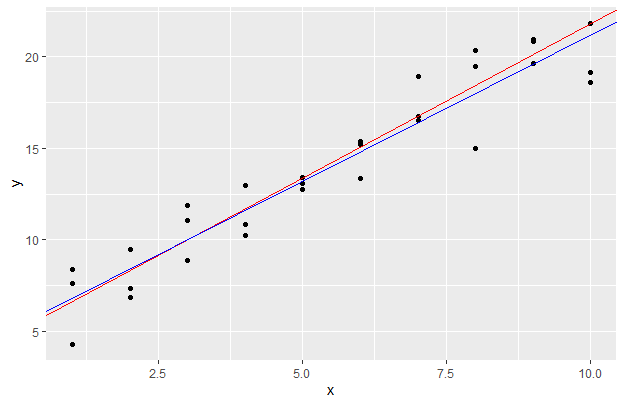
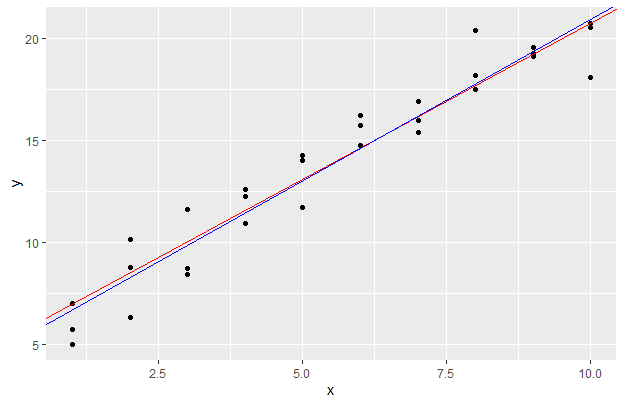
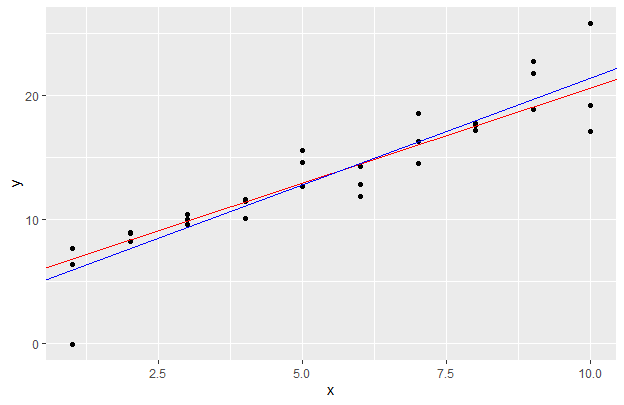
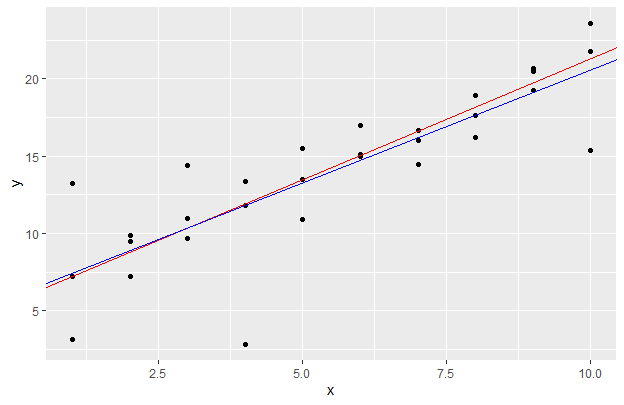
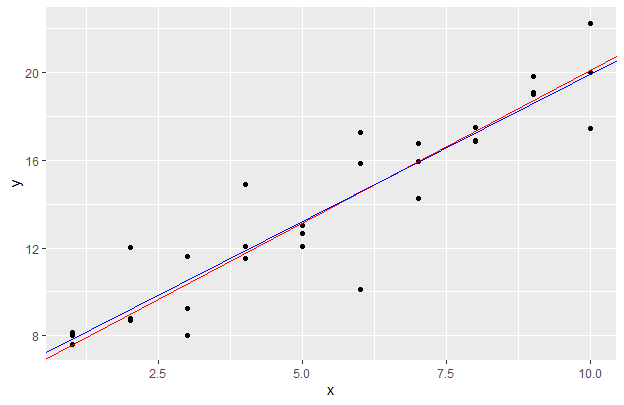
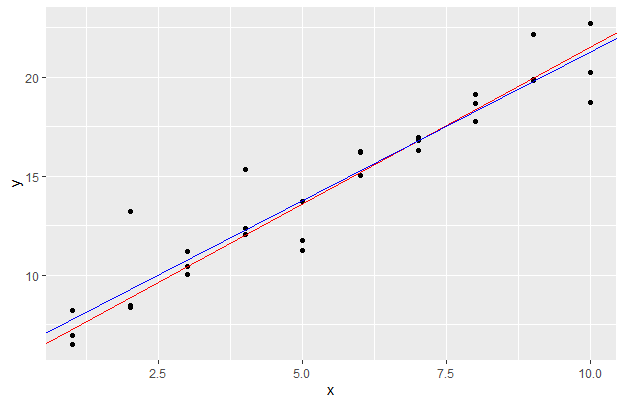
### (2) 23.2.1 Exercise 2

One way to make linear models more robust is to use a different distance measure. For example, instead of root-mean-squared (RMS) distance, you could use mean-absolute-deviation (MAD) distance:

Use optim() to fit this model to the simulated data above and visually compare it to the linear model, again examining at least 6 instances. You want both models to be drawn on top of each instance of the simulated data set. Discuss what you find.

*Hints*: make\_prediction() is not a built-in function; you will need to create your own linear function. Unlike Exercise 1, here you should only compare the models graphically; I don’t think the metric is meaningful for a model that uses the MAD distance measure.

# I understand the code below is not very optimized but I could not figure out how to make a proper looping function because I kept getting the error "data$y is closure variable".  
  
# The two models perform very similarly on each iteration; however the RMS model is more effected by outlier data than the MAD model is.   
  
  
  
  
sum <- 0  
  
repeat{  
 sum <- sum + 1  
   
 if (sum == 7){  
 break  
 }  
  
  
model1 <- function(a, data) {  
 a[1] + data$x \* a[2]  
}  
  
measure\_distanceMAD <- function(mod, data) {  
 diff <- data$y - model1(mod, data)  
 mean(abs(diff))  
}  
  
measure\_distanceRMS <- function(mod, data) {  
 diff <- data$y - model1(mod, data)  
 sqrt(mean(diff ^ 2))  
}  
  
sim1a <- tibble(  
 x = rep(1:10, each = 3),  
 y = x \* 1.5 + 6 + rt(length(x), df = 2)  
)  
  
best1 <- optim(c(0,0), measure\_distanceMAD, data = sim1a)  
best2 <- optim(c(0,0), measure\_distanceRMS, data = sim1a)  
  
plot1 <- ggplot(sim1a, aes(x, y)) +   
 geom\_point() +   
 geom\_abline(intercept = best1$par[1], slope = best1$par[2], color = "red")+  
 geom\_abline(intercept = best2$par[1],slope = best2$par[2], color = "blue")  
  
print(plot1)  
  
}



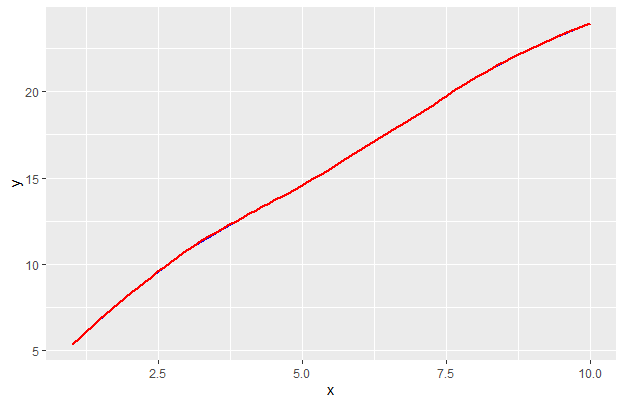
## 23.3.3 Exercises

### (3) 23.3.3 Exercise 1

Instead of using lm() to fit a straight line, you can use loess() to fit a smooth curve. Repeat the process of model fitting, grid generation, predictions, and visualization on sim1 using loess() instead of lm(). How does the result compare to geom\_smooth()?

# 'geom\_smooth()' uses 'loess()' as a default so they should look exactly the same.   
  
sim1\_loess <- loess(y ~ x, data = sim1)  
  
grid\_loess <- sim1 %>%  
 add\_predictions(sim1\_loess)  
  
sim1 <- sim1 %>%  
 add\_residuals(sim1\_loess, var = "resid\_loess") %>%  
 add\_predictions(sim1\_loess, var = "pred\_loess")  
  
plot\_sim1\_loess <-  
 ggplot(sim1, aes(x = x, y = y)) +  
 geom\_line(aes(x = x, y = pred), data = grid\_loess, colour = "blue")+  
 geom\_smooth(colour = "red", se = FALSE)  
plot\_sim1\_loess

## `geom\_smooth()` using method = 'loess' and formula 'y ~ x'



### (4) 23.3.3 Exercise 2

add\_predictions() is paired with gather\_predictions() and spread\_predictions(). Use both gather\_predictions() and spread\_predictions() to repeat the analysis with the two models used in problem (3) above. How do these three functions differ?

# The functions differ on how the data is stored. The gather predictions first selects one model then runs predictions on each model separately. Spread instead uses the same x and shows the result of each machine learning technique side by side. Personally, I see spread being the more useful and more effective calling of the predictions.  
  
  
# Assumed the second model is supposed to be 'lm()'  
  
sim1\_mod <- lm(y ~ x, data = sim1)  
grid <- sim1 %>%  
 data\_grid(x)  
  
grid %>%  
 gather\_predictions(sim1\_mod, sim1\_loess)

## # A tibble: 20 x 3  
## model x pred  
## <chr> <int> <dbl>  
## 1 sim1\_mod 1 6.27  
## 2 sim1\_mod 2 8.32  
## 3 sim1\_mod 3 10.4   
## 4 sim1\_mod 4 12.4   
## 5 sim1\_mod 5 14.5   
## 6 sim1\_mod 6 16.5   
## 7 sim1\_mod 7 18.6   
## 8 sim1\_mod 8 20.6   
## 9 sim1\_mod 9 22.7   
## 10 sim1\_mod 10 24.7   
## 11 sim1\_loess 1 5.34  
## 12 sim1\_loess 2 8.27  
## 13 sim1\_loess 3 10.8   
## 14 sim1\_loess 4 12.8   
## 15 sim1\_loess 5 14.6   
## 16 sim1\_loess 6 16.6   
## 17 sim1\_loess 7 18.7   
## 18 sim1\_loess 8 20.8   
## 19 sim1\_loess 9 22.6   
## 20 sim1\_loess 10 24.0

grid %>%  
 spread\_predictions(sim1\_mod, sim1\_loess)

## # A tibble: 10 x 3  
## x sim1\_mod sim1\_loess  
## <int> <dbl> <dbl>  
## 1 1 6.27 5.34  
## 2 2 8.32 8.27  
## 3 3 10.4 10.8   
## 4 4 12.4 12.8   
## 5 5 14.5 14.6   
## 6 6 16.5 16.6   
## 7 7 18.6 18.7   
## 8 8 20.6 20.8   
## 9 9 22.7 22.6   
## 10 10 24.7 24.0

### (5) 23.3.3 Exercise 3

What does geom\_ref\_line() do? What package does it come from? Why is displaying a reference line in plots showing residuals useful and important?

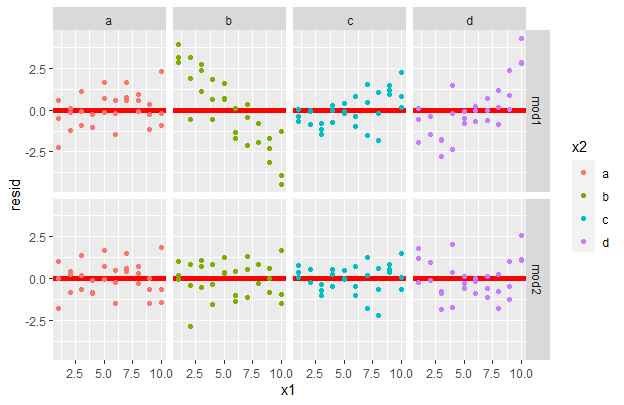
# It simply adds a reference line through all the graphs to easier see the correlation.  
  
mod1 <- lm(y ~ x1 + x2, data = sim3)  
mod2 <- lm(y ~ x1 \* x2, data = sim3)  
  
grid <- sim3 %>%  
 data\_grid(x1, x2) %>%  
 gather\_predictions(mod1, mod2)  
grid

## # A tibble: 80 x 4  
## model x1 x2 pred  
## <chr> <int> <fct> <dbl>  
## 1 mod1 1 a 1.67  
## 2 mod1 1 b 4.56  
## 3 mod1 1 c 6.48  
## 4 mod1 1 d 4.03  
## 5 mod1 2 a 1.48  
## 6 mod1 2 b 4.37  
## 7 mod1 2 c 6.28  
## 8 mod1 2 d 3.84  
## 9 mod1 3 a 1.28  
## 10 mod1 3 b 4.17  
## # ... with 70 more rows

sim3r <- sim3 %>%  
 gather\_residuals(mod1, mod2)  
sim3r

## # A tibble: 240 x 7  
## model x1 x2 rep y sd resid  
## <chr> <int> <fct> <int> <dbl> <dbl> <dbl>  
## 1 mod1 1 a 1 -0.571 2 -2.25   
## 2 mod1 1 a 2 1.18 2 -0.491   
## 3 mod1 1 a 3 2.24 2 0.562   
## 4 mod1 1 b 1 7.44 2 2.87   
## 5 mod1 1 b 2 8.52 2 3.96   
## 6 mod1 1 b 3 7.72 2 3.16   
## 7 mod1 1 c 1 6.51 2 0.0261  
## 8 mod1 1 c 2 5.79 2 -0.691   
## 9 mod1 1 c 3 6.07 2 -0.408   
## 10 mod1 1 d 1 2.11 2 -1.92   
## # ... with 230 more rows

ggplot(sim3r, aes(x = x1, y = resid, color = x2)) +  
 geom\_ref\_line(h = 0, colour = 'red') +  
 geom\_point() +  
 facet\_grid(rows = vars(model), cols = vars(x2))



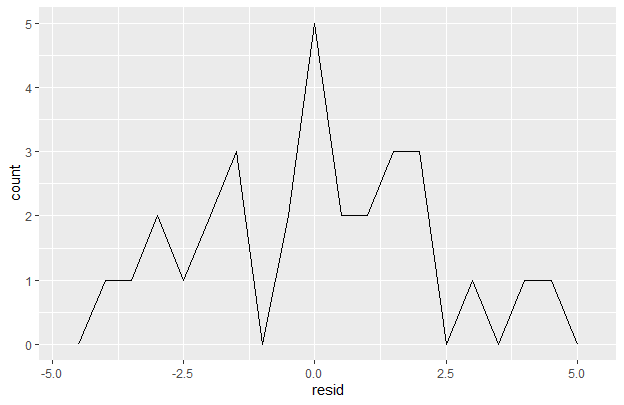
### (6) 23.3.3 Exercise 4

Why might you want to look at a frequency polygon of absolute residuals? What are the pros and cons compared to looking at the raw residuals?

# Since the direction of the residual does not matter, it allows the user to more effectively see the the values of the residual.  
  
  
sim1 <- sim1 %>%  
 add\_residuals(sim1\_mod)  
  
ggplot(sim1, aes(x = abs(resid))) +  
 geom\_freqpoly(binwidth = 0.5)



ggplot(sim1, aes(x = resid)) +  
 geom\_freqpoly(binwidth = 0.5)



## 23.4.5 Exercises

### (7) 23.4.5 Exercise 4

For sim4, which of mod1 and mod2 is better? The book author believes that mod2 does a slightly better job at removing patterns, but claims the effects are subtle. Examine the values for the two models, and develop a plot that attempts to support this claim. In the end, how well does your plot support the claim? How likely is it that the claim is true?

# I mean the R^2 score shows that both models are not particularly good at predicting, the mod2 has a better score but when it is that close to .50 then both models are just not good enough to be used. In the plot, it can be seen that the mod2 has a more condense residual spread, but still not enough to be viable.

mod1 <- lm(y ~ x1 + x2, data = sim4)  
mod2 <- lm(y ~ x1 \* x2, data = sim4)

grid <- sim4 %>%  
 data\_grid(x1,x2)  
grid <- grid %>%  
 gather\_predictions(mod1, mod2)  
  
sim4\_mods <- gather\_residuals(sim4, mod1, mod2)

m1r <- summary(mod1)$r.squared  
m2r <- summary(mod2)$r.squared  
print(sprintf("mod1 R^2: %.1f%%", m1r\*100))

## [1] "mod1 R^2: 50.5%"

print(sprintf("mod2 R^2: %.1f%%", m2r\*100))

## [1] "mod2 R^2: 52.2%"

print(sprintf("R^2\_mod2 - R^2\_mod1: %.1f%%", (m2r - m1r)\*100))

## [1] "R^2\_mod2 - R^2\_mod1: 1.7%"

ggplot(sim4\_mods, aes(x = x1, y = resid, color = x2)) +  
 geom\_ref\_line(h = 0, colour = 'red') +  
 geom\_point() +  
 facet\_grid(rows = vars(model), cols = vars(x2))

