Math 500 HW6

Antony Sikorski

Problem 1

(a)

We first use least squares to compute the line $T = a_0 + a_1 x$ and the parabola $T = \beta_0 + \beta_1 x + \beta_2 x^2$ that best fit the data. First, the line:

```
#Packages
suppressMessages(library(fields))

#Setup
year <- c(1960, 1964, 1968, 1972, 1976, 1980, 1984, 1988, 1992, 1996, 2000, 2004, 2008, 2012)
time <- c(10.32, 10.06, 9.95, 10.14, 10.06, 10.25, 9.99, 9.92, 9.96, 9.84, 9.87, 9.85, 9.69, 9.63)

x <- (year - 1960)/52

#First I'll do this with built in R linear regression -------------------- (Unnecessary)
A <- cbind(1, x)
objLine <- lm( time ~ A - 1)
objLineTable <- summary(objLine)$coefficients
a0True <- objLineTable[1]
a1True <- objLineTable[2]



#Now I'll do it using Least Squares ---------------------------------------

#solving for Q and R
qrDecomp <- qr(A)
Q <- qr.Q(qrDecomp)
R <- qr.R(qrDecomp)

#solving for our coefficients a0 and a1
alpha <- solve(R) %*% t(Q) %*% time
a0 <- alpha[1]
a1 <- alpha[2]
print(c(a0, a1))
```

```
## [1] 10.2217143 -0.5105714
```

```
#Testing to see if I did the problem correctly
#passed means same results for my method and the built-in R method
test.for.zero(a0, a0True)
```

```
## PASSED test at tolerance  1e-08
```

```
test.for.zero(a1, a1True)
```

```
## PASSED test at tolerance  1e-08
```

For a linear fit ($T = a_0 + a_1 x$), our equation ends up being approximately $T = 10.222 - 0.511x$.

Now, the quadratic fit:

```
#Built in R linear regression ---------------------------------------- (Unnecessary)
A <- cbind(1, x, x^2)
objCurve <- lm( time ~ A - 1)
objCurveTable <- summary(objCurve)$coefficients
B0True <- objCurveTable[1]
B1True <- objCurveTable[2]
B2True <- objCurveTable[3]



#Now we will use Least Squares: --------------------------------------
qrDecomp <- qr(A)
Q <- qr.Q(qrDecomp)
R <- qr.R(qrDecomp)

#solving for our coefficients B0, B1, and B2
Beta <- solve(R) %*% t(Q) %*% time
B0 <- Beta[1]
B1 <- Beta[2]
B2 <- Beta[3]
print(c(B0,B1,B2))
```

```
## [1] 10.1668929 -0.1542321 -0.3563393
```

```
#testing to see I did the problem correctly
#passed means same results for my method and the built-in R method
test.for.zero(B0True, B0)
```

```
## PASSED test at tolerance  1e-08
```

```
test.for.zero(B1True, B1)
```

```
## PASSED test at tolerance  1e-08
```

```
test.for.zero(B2True, B2)
```

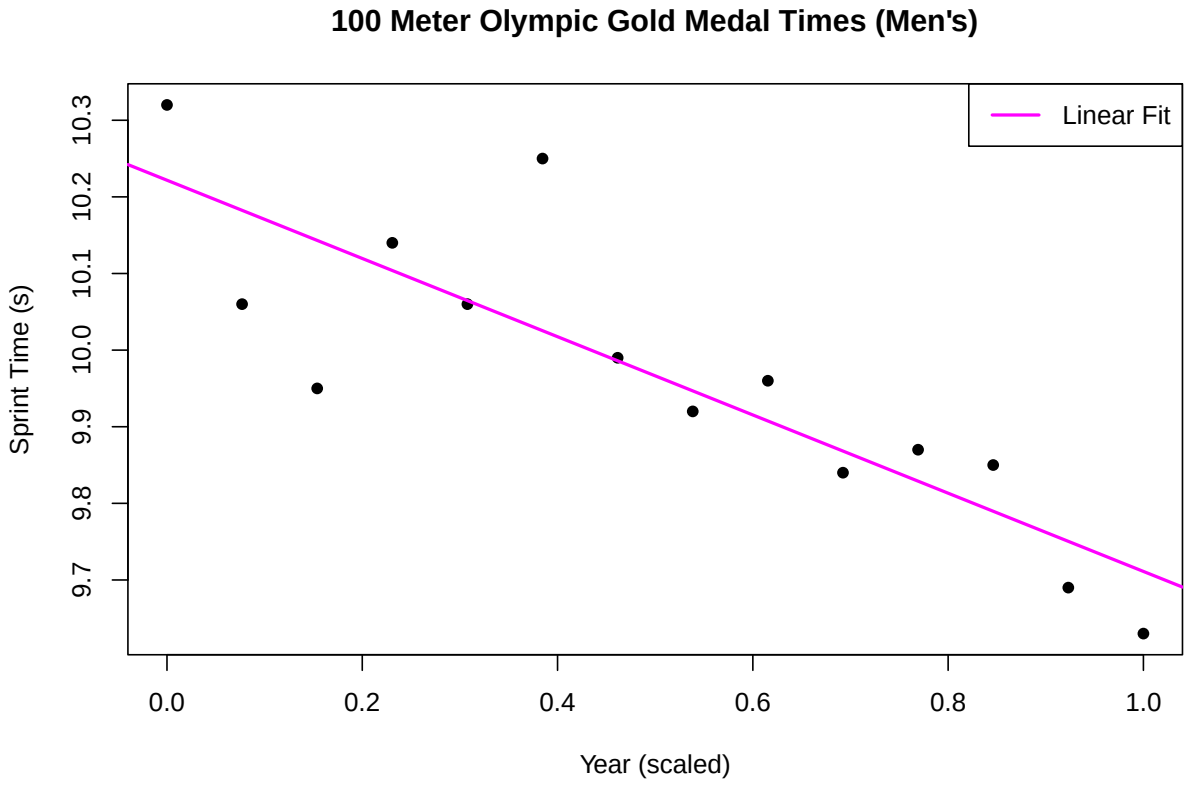
```
## PASSED test at tolerance  1e-08
```

For a quadratic fit ($T = \beta_0 + \beta_1 x + \beta_2 x^2$), our equation ends up being approximately $T = 10.167 - 0.154x + 0.356x^2$.

(b)

Now we plot both the linear and the quadratic fits. First, the linear fit:

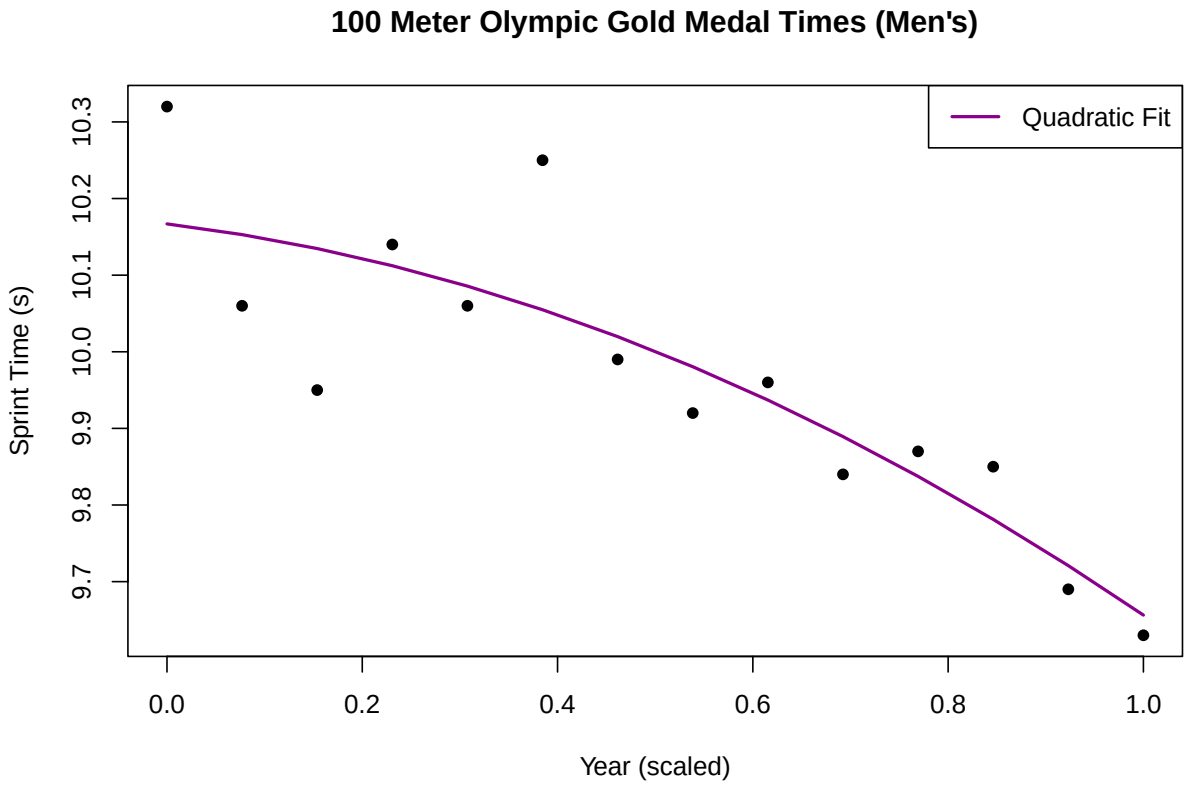
```
#Linear Fit Plot
plot(x, time, main = "100 Meter Olympic Gold Medal Times (Men's)",
     xlab = "Year (scaled)", ylab = "Sprint Time (s)", pch = 16)
abline(a0, a1, lwd = 2, col = "magenta")
legend("topright", legend = c("Linear Fit"), col = c("magenta"), lwd = 2)
```



Now, the quadratic fit:

```
#Curve
parabola <- A %*%Beta

#Quadratic Fit Plot
plot(x, time, main = "100 Meter Olympic Gold Medal Times (Men's)",
     xlab = "Year (scaled)", ylab = "Sprint Time (s)", pch = 16)
lines(x, parabola, lwd = 2, col = "darkmagenta")
legend("topright", legend = c("Quadratic Fit"), col = c("darkmagenta"), lwd = 2)
```



(c)

Now, we predict the 2020 (now 2021) gold medal sprint times for the men's 100 yard dash using both of our models. I'll just do both years because I'm not sure which one the answer key will have (although I suspect 2020):

```
x2020 <- (2020 - 1960)/52
x2021 <- (2021 - 1960)/52

#im well aware that I use too many parens
#trust issues i guess..

#functions for predicting
linePredict <- function(yr){
  sprintTime = a0 + (a1*yr)
  return(sprintTime)
}

parabolaPredict <- function(yr){
  sprintTime = B0 + (B1*yr) + (B2 * (yr^2))
  return(sprintTime)
}

#actual predictions
print(c(linePredict(x2020),linePredict(x2021)))
```

```
## [1] 9.632593 9.622775
```

```
print(c(parabolaPredict(x2020),parabolaPredict(x2021)))
```

```
## [1] 9.514516 9.495605
```

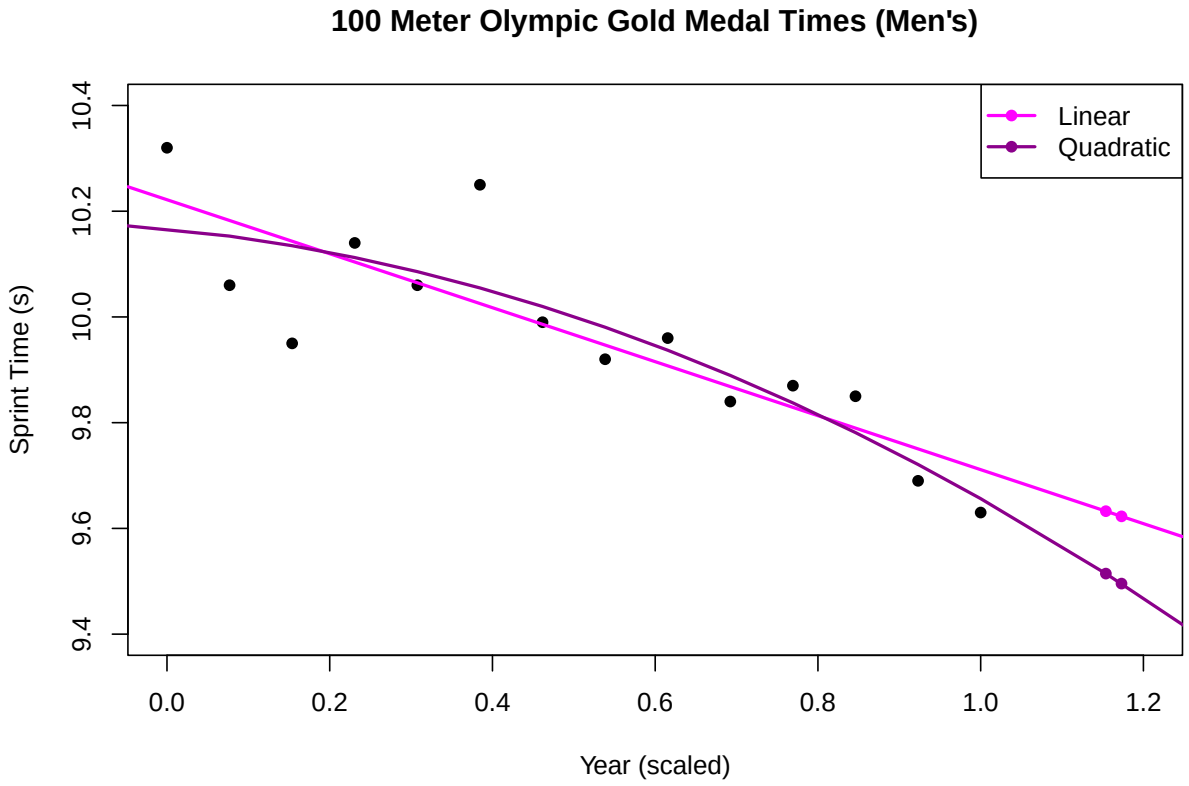
The linear prediction for the year of 2020 is 9.633, and for the year of 2021 it is 9.623. The quadratic prediction for the year of 2020 is 9.515, and for the year of 2021 it is 9.496. Here is a nice graph to put it all together:

```
#need to do some adjusting so the parabola curve takes up the whole graph window
#the numbers are wrong but none of the math is (ignore this bit)
year1 <- c(1956, 1964, 1968, 1972, 1976, 1980, 1984, 1988, 1992, 1996, 2000, 2004, 2008, 2012, 2020, 20
x1 <- (year1 - 1960)/52
A1 <- cbind(1, x1, x1^2)
parabola1 <- A1%*%Beta

#plotting
plot(x, time, main = "100 Meter Olympic Gold Medal Times (Men's)",
     xlab = "Year (scaled)", ylab = "Sprint Time (s)", pch = 16,
     ylim = c(9.4, 10.4), xlim = c(0, 1.2))
abline(a0, a1, lwd = 2, col = "magenta")
lines(x1, parabola1, lwd = 2, col = "darkmagenta")
legend("topright", legend = c("Linear", "Quadratic"), col = c("magenta", "darkmagenta"), lwd = 2, pch =

#prediction points
```

```
points(x2020, linePredict(x2020), col = "magenta", pch = 16)
points(x2021, linePredict(x2021), col = "magenta", pch = 16)
points(x2020, parabolaPredict(x2020), col = "darkmagenta", pch = 16)
points(x2021, parabolaPredict(x2021), col = "darkmagenta", pch = 16)
```



Problem 2

(a)

We now rank the football teams (1-5) by their winning percentage:

```
#setup
#manually entering data
win <- c(1, 3, 1, 2, 4, 3, 2, 3, 3, 5)
loss <- c(2, 5, 5, 5, 1, 4, 4, 2, 1, 4)
diff <- c(7, 1, 20, 7, 24, 4, 10, 12, 3, 1)


#creating A matrix
A <- matrix(0, nrow = length(win), ncol = max(win))

for (i in 1:dim(A)[1]){
  A[i, win[i]] = 1
  A[i, loss[i]] = -1
}



#Ranking by Win %:
WinPercentdf <- data.frame(matrix(ncol = 3))
colnames(WinPercentdf) <- c("Team", "Win %", "Rank")

for (i in 1:dim(A)[2]){
  WinPercentdf[i,1] = paste0("Team ", i)
  WinPercentdf[i,2] = length(A[,i][A[,i] == 1])/sum(abs(A[,i])) * 100
}

WinPercentdf <- WinPercentdf[order(WinPercentdf[,2], decreasing = TRUE),]
WinPercentdf[,3] <- 1:5
print(WinPercentdf, row.names = FALSE)
```

```
##    Team Win % Rank
##  Team 3   100    1
##  Team 1    50    2
##  Team 2    50    3
##  Team 4    25    4
##  Team 5    25    5
```

(b)

Now we rank the teams by their Massey index:

```
#Setting up matrices
AtA <- t(A) %*% A
Atb <- t(A) %*% diff
AtA[5,] <- c(1,1,1,1,1)
Atb[5,] <- 0

#solving AtAm = Atb
m <- solve(AtA)%*%Atb
m <- round(m, 4)
Team <- c("Team 1", "Team 2", "Team 3", "Team 4", "Team 5")
Rank <- c()

#producing results
Masseydf <- data.frame(cbind(Team, m, Rank))
Masseydf <- rbind(Masseydf[3,],Masseydf[4,],Masseydf[1,],Masseydf[2,],Masseydf[5,])
Masseydf[,3] <- 1:5
colnames(Masseydf) <- c("Team", "Massey Index", "Rank")
print(Masseydf, row.names = FALSE)
```

```
##    Team Massey Index Rank
##  Team 3            4    1
##  Team 4          1.8    2
##  Team 1            0    3
##  Team 2         -0.4    4
##  Team 5         -5.4    5
```

(c)

We fiddle with the rankings, and change the last game so that Team 5 beats Team 4 by 20 points rather than by 1 point. Here is the effect on the Massey rankings:

```
diff1 <- c(7, 1, 20, 7, 24, 4, 10, 12, 3, 20)

#Recomputing:
AtA <- t(A) %*% A
Atb <- t(A) %*% diff1
AtA[5,] <- c(1,1,1,1,1)
Atb[5,] <- 0

#solving AtAm = Atb
m <- solve(AtA)%*%Atb
m <- round(m, 4)
Team <- c("Team 1", "Team 2", "Team 3", "Team 4", "Team 5")
Rank <- c()

Masseydf <- data.frame(cbind(Team, m, Rank))
Masseydf <- rbind(Masseydf[3,],Masseydf[1,],Masseydf[2,],Masseydf[5,],Masseydf[4,])


Masseydf[,3] <- 1:5
colnames(Masseydf) <- c("Team", "Massey Index", "Rank")
print(Masseydf, row.names = FALSE)
```

```
##    Team Massey Index Rank
##  Team 3            4    1
##  Team 1            0    2
##  Team 2         -0.4    3
##  Team 5         -1.6    4
##  Team 4           -2    5
```

We fiddle with the score of the last game until Team 5 receives the highest Massey index out of all of the teams. We find that Team 5 needs to beat Team 4 in the last game by 49 points in order to be at the top of the rankings. Proof:

```
#just playing with the last value
lastVal <- 49
diff2 <- c(7, 1, 20, 7, 24, 4, 10, 12, 3, lastVal)
#Recomputing:
AtA <- t(A) %*% A
Atb <- t(A) %*% diff2
AtA[5,] <- c(1,1,1,1,1)
Atb[5,] <- 0

#solving AtAm = Atb
m <- solve(AtA)%*%Atb
m <- round(m, 4)
m <- cbind(m, Team)
m
```

```
##             Team
## [1,] "0"    "Team 1"
## [2,] "-0.4" "Team 2"
## [3,] "4"    "Team 3"
## [4,] "-7.8" "Team 4"
## [5,] "4.2"  "Team 5"
```

Looking at our m column vector, it is clear that Team 5 now has the highest Massey index!