Question 1

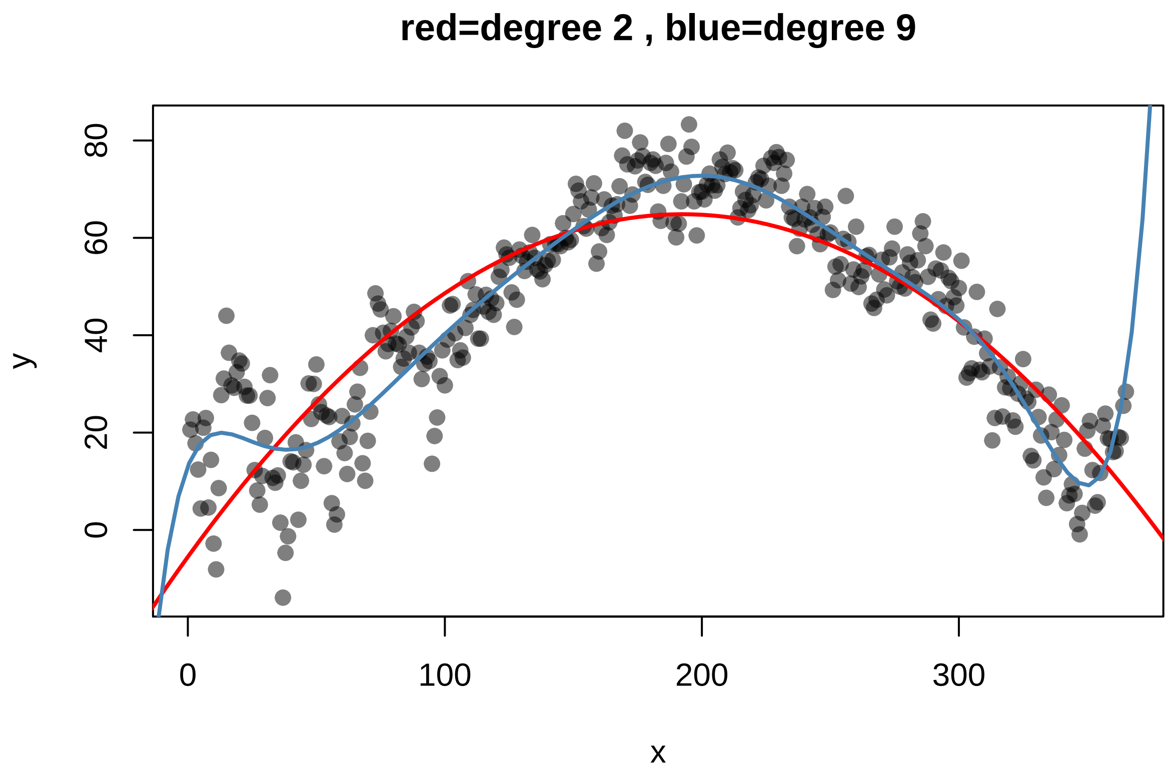
Jin Barai

```r
ottawa1995 <- read.csv("./ottawaTemp1995.csv")
ottawaOther <- read.csv("./ottawaOtherYears.csv")
n <- 365
x <- 1:n
temp1995.data <- data.frame(x=1:n, y=ottawa1995$Temp[1:n])
```

(a)

Scatter plot of the data and overlay fitted polynomials with degrees 2 and 9 to the data:

```r
muhat2 <- getmuhat(temp1995.data, 2)
muhat9 <- getmuhat(temp1995.data, 9)

xlim <- extendrange(temp1995.data[x,])

plot(temp1995.data,
     pch=19, col= adjustcolor("black", 0.5))
curve(muhat2, from = xlim[1], to = xlim[2],
      add = TRUE, col="red", lwd=2)
curve(muhat9, from = xlim[1], to = xlim[2],
      add = TRUE, col="steelblue", lwd=2)
title(main="red=degree 2 , blue=degree 9")
```



(b)

Generating m = 25 samples of size n = 50 and fitting polynomials of degree 2 and 9 to every sample

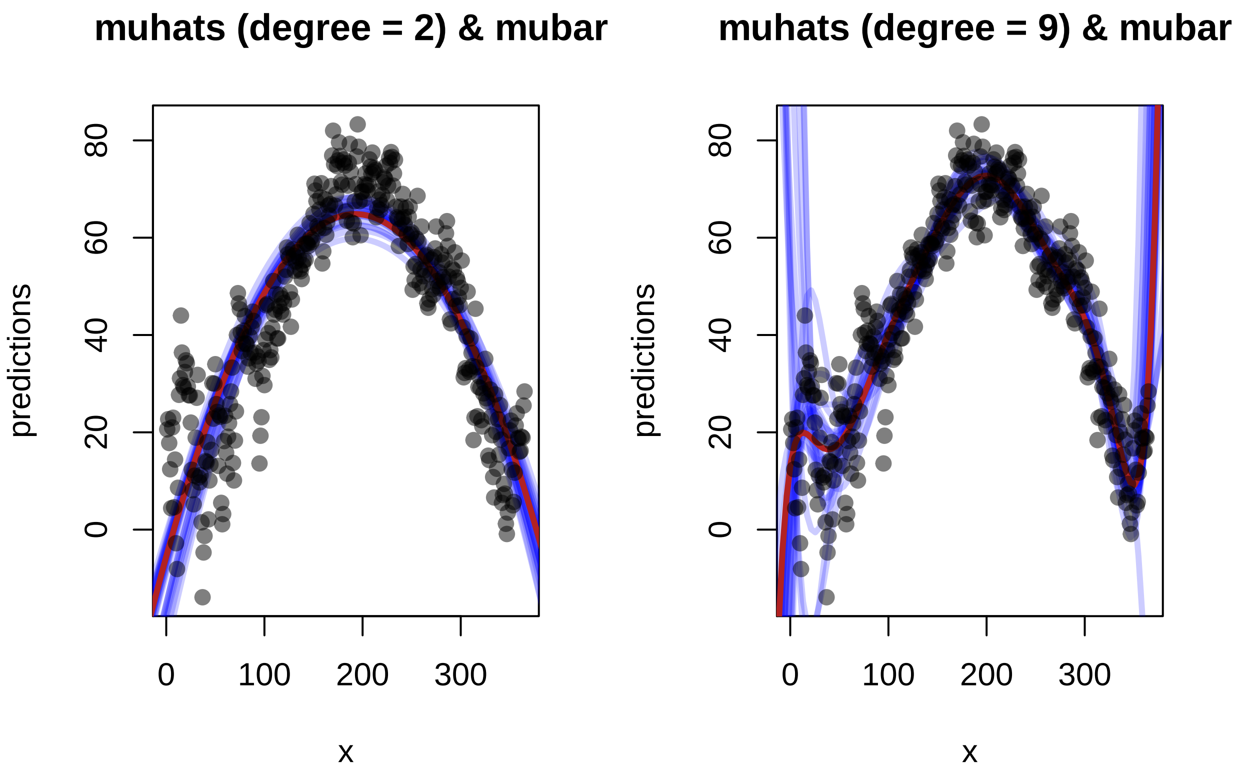
```
getSampleComp <- function(pop, size, replace=FALSE) {
  N <- dim(pop)[1]
  samp <- rep(FALSE, N)
  samp[sample(1:N, size, replace = replace)] <- TRUE
  samp
}


### This function will return a data frame containing
### only two variates, an x and a y
getXYSample <- function(xvarname, yvarname, samp, pop) {
  sampData <- pop[samp, c(xvarname, yvarname)]
  names(sampData) <- c("x", "y")
  sampData
}
```

```
N_S <- 25
set.seed(341)   # for reproducibility

n= 50
samps <- lapply(1:N_S, FUN= function(i){getSampleComp(temp1995.data, n)})
```

```
Ssamples <- lapply(samps, FUN= function(Si){getXYSample("x", "y", Si, temp1995.data)})
Tsamples <- lapply(samps, FUN= function(Si){getXYSample("x", "y", !Si, temp1995.data)})

muhats2 <- lapply(Ssamples, getmuhat, complexity = 2)
#getmubar(muhats2)

muhats9 <- lapply(Ssamples, getmuhat, complexity = 9)
#mubar10 <- getmubar(muhats10)
```

(c)

```
par(mfrow=c(1,2))

xvals <- seq(xlim[1], xlim[2], length.out = 200)
plot(temp1995.data,
     pch=19, type='n',
     xlab="x", ylab="predictions",
     main= " muhats (degree = 2) & mubar")


for (i in 1:N_S) {
  curveFn <- muhats2[[i]]
  curve(curveFn, from = xlim[1], to = xlim[2], add=TRUE, col=adjustcolor("blue", 0.2), lwd=3, lty=(1))
}

curve(muhat2,  from = xlim[1], to = xlim[2],
      add=TRUE, col="firebrick", lwd=3)

points(temp1995.data,
     pch=19, col= adjustcolor("black", 0.5))


plot(temp1995.data,
     pch=19, type='n',
     xlab="x", ylab="predictions",
     main= " muhats (degree = 9) & mubar")

for (i in 1:N_S) {
  curveFn <- muhats9[[i]]
  curve(curveFn, xlim[1], xlim[2], add=TRUE, col=adjustcolor("blue", 0.2), lwd=3, lty=1)
}

curve(muhat9, xlim[1], xlim[2], add=TRUE, col="firebrick", lwd=3)

points(temp1995.data,
     pch=19, col= adjustcolor("black", 0.5))
```



(d)

```
getmubar <- function(muhats) {
  function(x) {
    Ans <- sapply(muhats, FUN=function(muhat){muhat(x)})
    apply(Ans, MARGIN=1, FUN=mean)
  }
}

ave_y_mu_sq <- function(sample, predfun, na.rm = TRUE){
  mean((sample$y - predfun(sample$x))^2, na.rm = na.rm)
}

###########

ave_mu_mu_sq <- function(predfun1, predfun2, x, na.rm = TRUE){
  mean((predfun1(x) - predfun2(x))^2, na.rm = na.rm)
}

###########


var_mutilde <- function(Ssamples, Tsamples, complexity){
  ## get the predictor function for every sample S
  muhats <- lapply(Ssamples,
```

```
                   FUN=function(sample){
                     getmuhat(sample, complexity)
                   }
  )
  ## get the average of these, mubar
  mubar <- getmubar(muhats)

  ## average over all samples S
  N_S <- length(Ssamples)
  mean(sapply(1:N_S,
              FUN=function(j){
                ## get muhat based on sample S_j
                muhat <- muhats[[j]]
                ## average over (x_i,y_i) in a
                ## single sample T_j the squares
                ## (y - muhat(x))^2
                T_j <- Tsamples[[j]]
                ave_mu_mu_sq(muhat, mubar, T_j$x)
              }
  )
  )
}
```

Sampling variability of the function of polynomial with degree 2

```
var_mutilde(Ssamples, Tsamples, complexity=2)
```

```
## [1] 8.783346
```

Sampling variability of the function of polynomial with degree 9

```
var_mutilde(Ssamples, Tsamples, complexity=9)
```

```
## [1] 174.5025
```

(e)

```
bias2_mutilde <- function(Ssamples, Tsamples, mu, complexity){
  ## get the predictor function for every sample S
  muhats <- lapply(Ssamples,
                   FUN=function(sample) getmuhat(sample, complexity)
  )
  ## get the average of these, mubar
  mubar <- getmubar(muhats)

  ## average over all samples S
  N_S <- length(Ssamples)
  mean(sapply(1:N_S,
              FUN=function(j){
                ## average over (x_i,y_i) in a
                ## single sample T_j the squares
```

```
                ## (y - muhat(x))^2
                T_j <- Tsamples[[j]]
                ave_mu_mu_sq(mubar, mu, T_j$x)
              }
  )
  )
}
```

Squared bias of the polynomial with degree 2:

```
muhat = getmuFun(temp1995.data, "x", 'y')

bias2_mutilde(Ssamples, Tsamples, muhat, complexity=2)
```

```
## [1] 96.34388
```

Squared bias of the polynomial with degree 9:

```
bias2_mutilde(Ssamples, Tsamples, muhat, complexity=9)
```

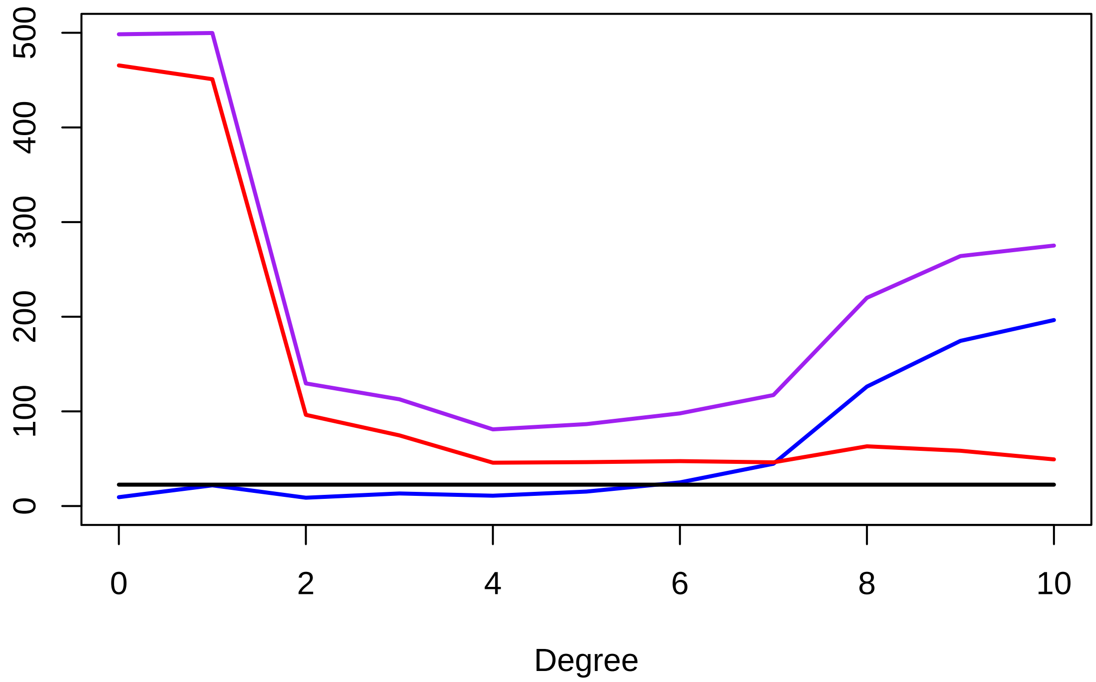
```
## [1] 58.41723
```

(f)

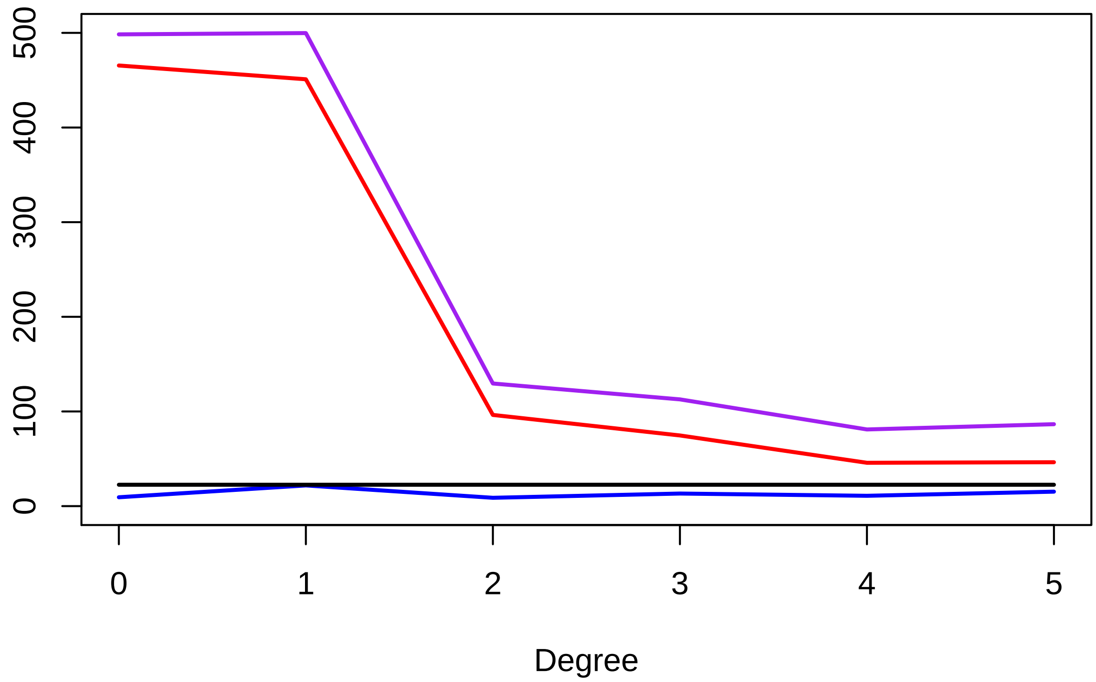
```
complexities <- 0:10

apse_vals <-  sapply(complexities,
                     FUN = function(complexity){
                       apse_all(Ssamples, Tsamples,
                                complexity = complexity, mu = muhat)
                     }
)

# Print out the results
t(rbind(complexities, apse=round(apse_vals,5)))
```

```
##       complexities      apse var_mutilde     bias2    var_y
##  [1,]            0 498.43265     9.36203 465.53831 22.57973
##  [2,]            1 499.77607    21.83982 450.97375 22.57973
##  [3,]            2 129.54484     8.78335  96.34388 22.57973
##  [4,]            3 112.78430    13.27332  74.63924 22.57973
##  [5,]            4  80.99503    10.88906  45.76678 22.57973
##  [6,]            5  86.55462    15.27438  46.36353 22.57973
##  [7,]            6  97.89267    24.97699  47.45176 22.57973
##  [8,]            7 117.23912    44.64995  46.17284 22.57973
##  [9,]            8 220.00011   126.18439  63.09053 22.57973
## [10,]            9 264.09327   174.50248  58.41723 22.57973
## [11,]           10 275.16609   196.49468  49.28097 22.57973
```

```r
plot( complexities, apse_vals[1,], xlab="Degree", ylab="", type='l', ylim=c(0, 500), col="purple", lwd=
lines(complexities, apse_vals[2,], col="blue", lwd=2 )
lines(complexities, apse_vals[3,], col="red", lwd=2)
lines(complexities, apse_vals[4,], col="black", lwd=2)
```



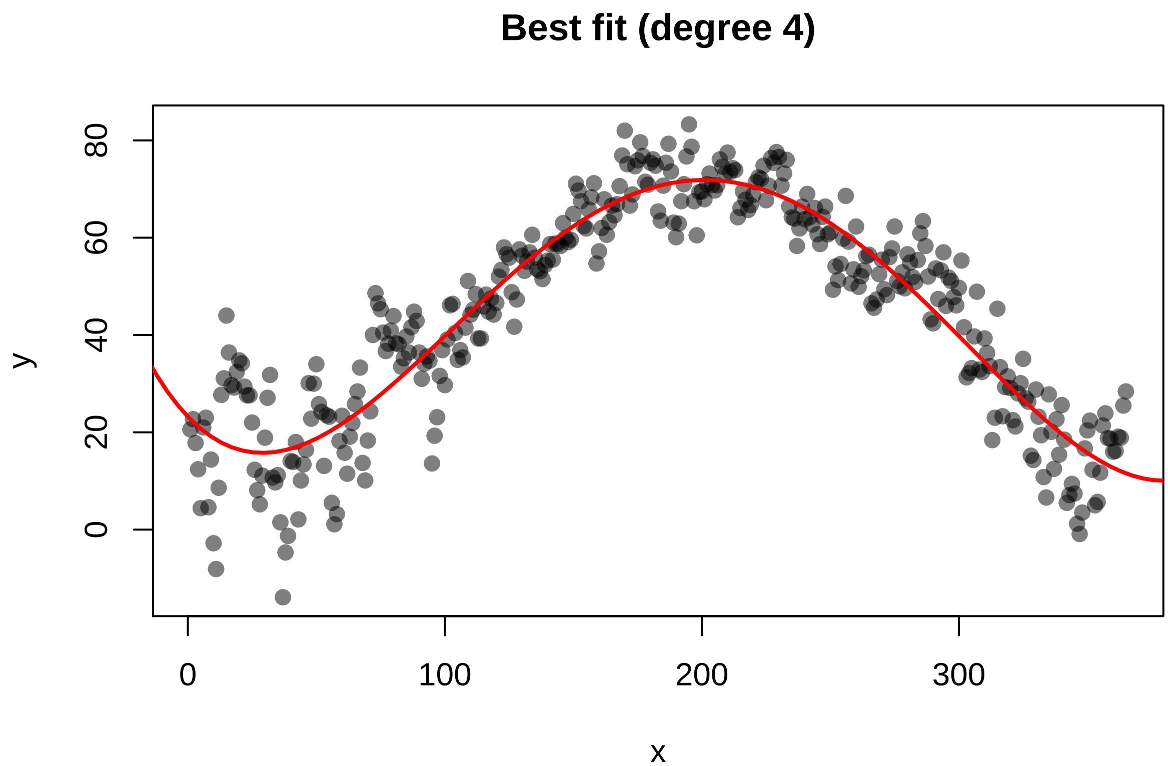

```r
# The increase in apse is too sharp in higher complexities. Let's zoom in a bit
zoom = 0:6
plot( complexities[zoom], apse_vals[1, zoom], xlab="Degree", ylab="", type='l', ylim=c(0, 500), col="pu
lines(complexities[zoom], apse_vals[2, zoom], col="blue", lwd=2 )
lines(complexities[zoom], apse_vals[3, zoom], col="red", lwd=2)
lines(complexities[zoom], apse_vals[4, zoom], col="black", lwd=2)
```



Conclusion:

- The polynomial with degree 4 has the lowest APSE
- The APSE first decreases with an increase in the degree but then after degree 5, it increases again
- The bias starts off high and then decreases and hovers about values 65 - 70s

```
# Code for lowest ASPE
muhat3 <- getmuhat(temp1995.data, 4)

xlim <- extendrange(temp1995.data[x,])

plot(temp1995.data,
     pch=19, col= adjustcolor("black", 0.5),
     main="Best fit (degree 4)")
curve(muhat3, from = xlim[1], to = xlim[2],
      add = TRUE, col="red", lwd=2)
```



The visual assessment confirms a decent fit of degree 4 polynomial

(g)

(i)

Function that creates the k-fold samples from a given population.

```
sample.muFun = getmuFun(temp1995.data, "x", 'y')
sample.kfold <- function(k=NULL, pop=NULL, xvarname=NULL, yvarname=NULL ) {

  N = nrow(pop)
  kset = rep_len(1:k, N)
  kset = sample(kset)

  samps = list()
  for (i in 1:k) {
    samps[[i]] = logical(N)
    samps[[i]][kset != i] = TRUE
  }

Ssamples <- lapply(samps,
                   FUN= function(Si){getXYSample(xvarname, yvarname, Si, pop)})

Tsamples <- lapply(samps,
```

```r
                    FUN= function(Si){getXYSample(xvarname, yvarname, !Si, pop)})

    list(Ssamples=Ssamples,Tsamples=Tsamples)
}
```

(ii)

```r
kfold.samples = sample.kfold(k=5, pop=temp1995.data, "x", "y")
apse_all(kfold.samples$Ssamples, kfold.samples$Tsamples, complexity = 2, mu = sample.muFun)
```

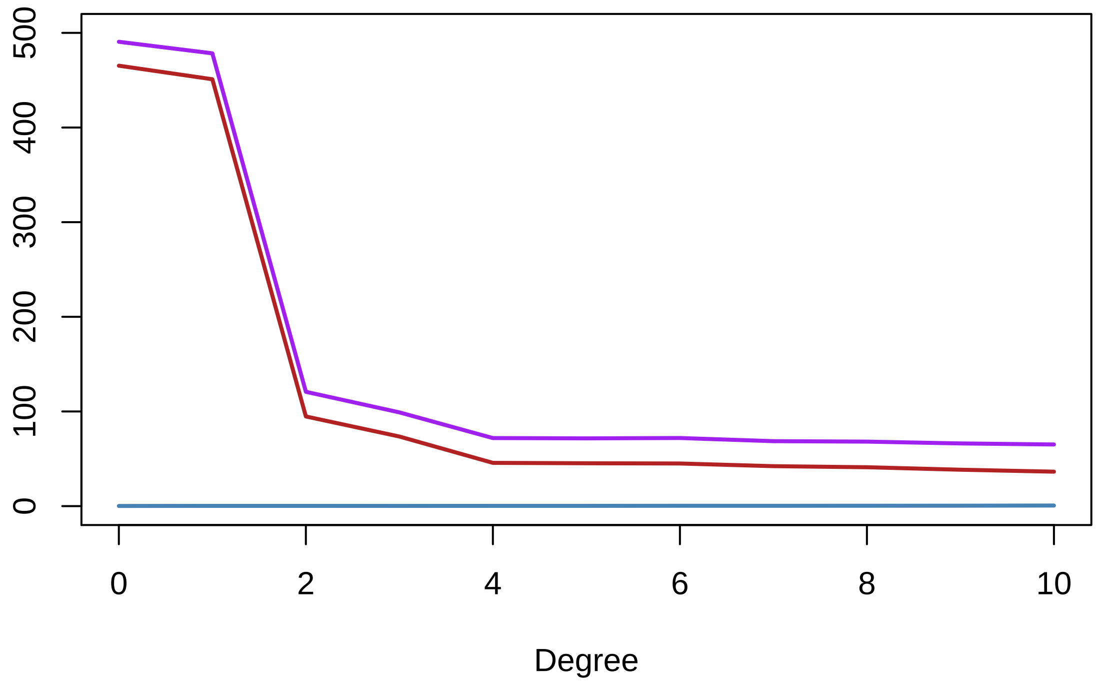
```
##        apse var_mutilde       bias2       var_y
## 118.1101478   0.1475614  94.7992658  22.5318493
```

(iii)

```r
complexities <- 0:10
kfold2.samples = sample.kfold(k=10, pop=temp1995.data, "x", "y")

apse_vals2 <-sapply(complexities,
      FUN = function(complexity){
          apse_all(kfold2.samples$Ssamples, kfold2.samples$Tsamples,
              complexity = complexity, mu = muhat) })

# Print out the results
t(rbind(complexities, apse=round(apse_vals2,5)))
```

```
##       complexities      apse var_mutilde     bias2    var_y
##  [1,]            0 490.58824     0.15333 465.33989 22.52661
##  [2,]            1 478.37128     0.25451 451.00295 22.52661
##  [3,]            2 120.77728     0.23220  94.74057 22.52661
##  [4,]            3  99.02791     0.21046  73.53717 22.52661
##  [5,]            4  71.88955     0.26744  45.70190 22.52661
##  [6,]            5  71.59213     0.28205  45.28162 22.52661
##  [7,]            6  71.94396     0.36102  45.01687 22.52661
##  [8,]            7  68.60484     0.31979  42.19725 22.52661
##  [9,]            8  68.15029     0.38219  41.06213 22.52661
## [10,]            9  66.22440     0.45504  38.44444 22.52661
## [11,]           10  65.11131     0.59793  36.38073 22.52661
```

```r
plot(complexities, apse_vals2[3,], xlab="Degree", ylab="", type='l', ylim=c(0, 500), col="firebrick", lw
lines(complexities, apse_vals2[2,], xlab="Degree", ylab="", col="steelblue", lwd=2 )
lines(complexities, apse_vals2[1,], col="purple", lwd=2)
```



Conclusion:

- As we can see from above table and graph the APSE sharply decreases after degree 1
- The polynomial of degree 10 has the lowest APSE so we would pick that
- The bias also decreases sharply after degree 2 and gradually after degree 4

(h)

```
ottawa <- read.csv("./ottawaOtherYears.csv")
n <- 6732
x <- 1:n
temp.data <- data.frame(x=1:n, y=ottawa$Temp[1:n])

complexities <- 0:10
kfold2.samples = sample.kfold(k=10, pop=temp.data, "x", "y")


sampsTesting <- lapply(1:N_S, FUN= function(i){getSampleComp(temp.data, n)})
Tsamples <- lapply(sampsTesting, FUN= function(Si){getXYSample("x", "y", Si, temp.data)})
apse_vals <- sapply(complexities,
FUN = function(complexity){
apse_all(Ssamples, Tsamples,
complexity = complexity, mu = muhat)
}
)
```

```r
# Print out the results
t(rbind(complexities, apse=round(apse_vals,5)))
```

```
##       complexities         apse  var_mutilde       bias2    var_y
##  [1,]            0 4.556035e+02 9.362030e+00 2.385854e+02 667.2422
##  [2,]            1 3.099045e+04 1.532483e+04 1.882993e+04 667.2422
##  [3,]            2 1.411608e+09 1.166388e+07 1.399079e+09 667.2422
##  [4,]            3 4.259574e+11 5.225656e+10 3.736883e+11 667.2422
##  [5,]            4 1.403087e+15 7.352485e+13 1.329563e+15 667.2422
##  [6,]            5 5.468071e+17 5.440241e+17 2.782994e+15 667.2422
##  [7,]            6 2.679036e+21 2.043079e+21 6.359573e+20 667.2422
##  [8,]            7 2.804165e+25 2.333040e+25 4.711248e+24 667.2422
##  [9,]            8 2.550690e+29 1.501764e+29 1.048926e+29 667.2422
## [10,]            9 6.039813e+32 4.262297e+32 1.777516e+32 667.2422
## [11,]           10 1.937486e+36 9.728032e+35 9.646831e+35 667.2422
```

```r
muhat3 <- getmuhat(temp1995.data, 4)
xlim <- extendrange(temp1995.data[x,])
plot(temp.data,
pch=19, col= adjustcolor("black", 0.5),
main="Best fit (degree 4)")
curve(muhat3, from = xlim[1], to = xlim[2],
add = TRUE, col="red", lwd=2)
```