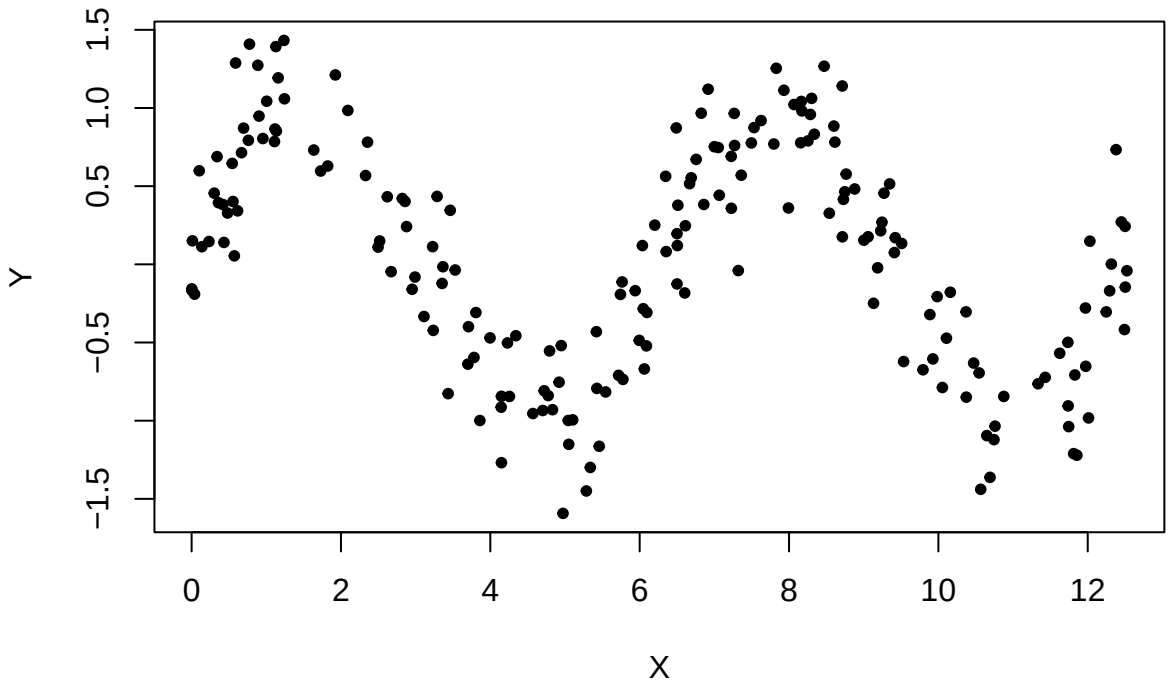
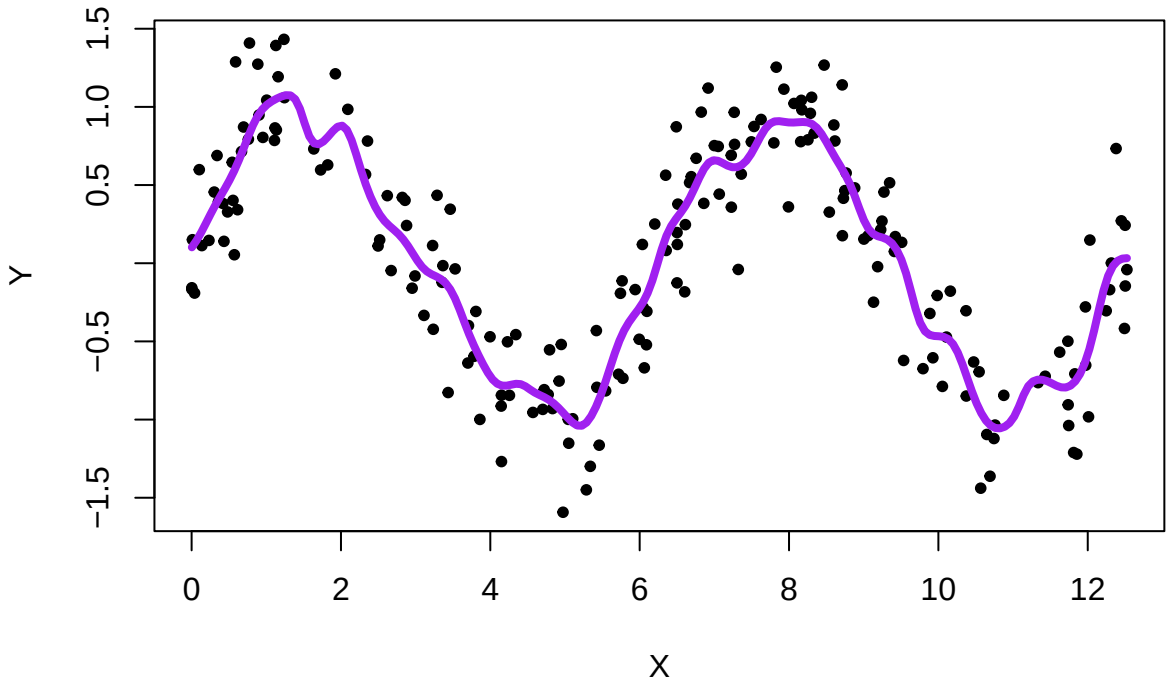
[ESC 21FALL] Homework 1

```
X = sort(runif(200, min=0, max=4*pi)) # generate random number btw 0~4*pi
Y = sin(X) + rnorm(200, sd=0.3)       # add noise to sin function
plot(X, Y, pch=20)                    # draw scatterplot
```



First, let's see how `ksmooth` function works in default R.

```
Kreg = ksmooth(x=X, y=Y, kernel="normal", bandwidth=0.5)
plot(X, Y, pch=20)
lines(Kreg, lwd=4, col="purple")
```



Part 2. Optimal Bandwidth (refer to hw description in `README.md`)

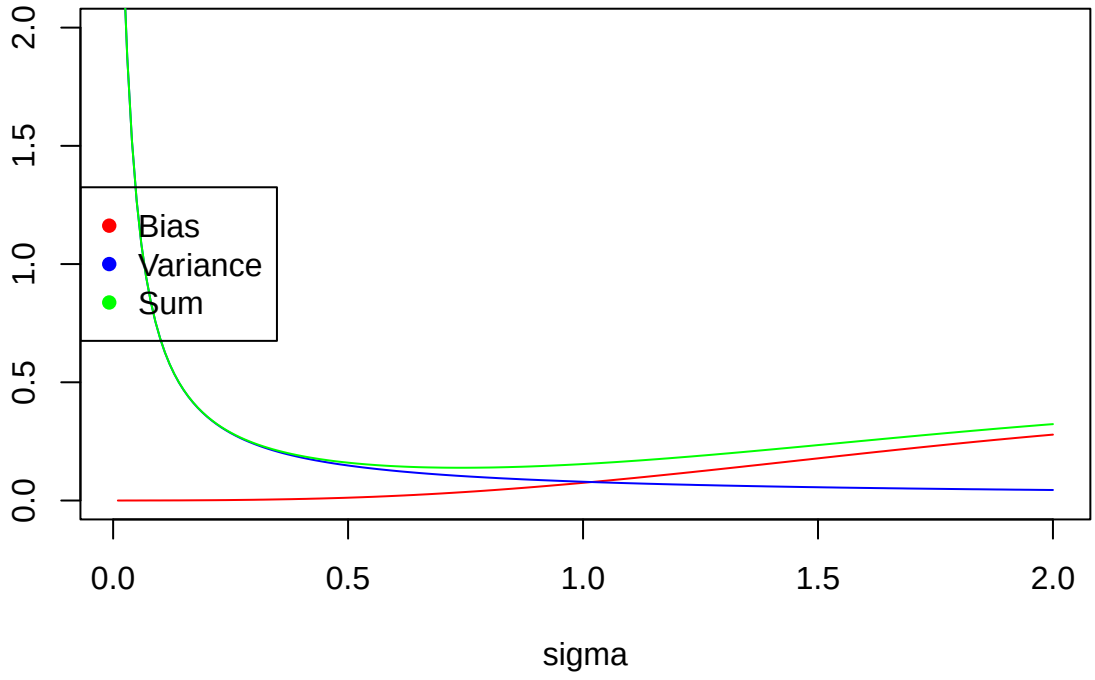
(a) Using a Gaussian kernel ϕ_σ, plot squared bias, variance, and their sum for $\sigma =$`seq(from = 0.01, to = 2, by = 0.01)`. Print the optimal choice for σ.

```
source('home1-part2-data.R')
```

(This process takes some time. . . !)

```
# Initialization
sigma = seq(from = 0.01, to = 2, by = 0.01)
n = length(sigma)
squared_norm <- function(x) sum(x^2)  # will be used for computing bias^2

normal <- function(x, mean, sigma) (1/(sqrt(2*pi)*sigma))*exp(-0.5*((x-mean)/ sigma)^2)

# initialize empty list to store values
bias = numeric(n)          # stores bias^2
variance = numeric(n)
summation = numeric(n)

for (k in 1:n) {
  # W : weight matrix (kernel function value)
  # make sure to include normalizing part!
  W = matrix(nrow = n, ncol = n)
  for (i in 1:n) {         # move filter (kernel) through query point
    for (j in 1:n) {       # local neighborhood (all data due to Gaussian kernel)
```

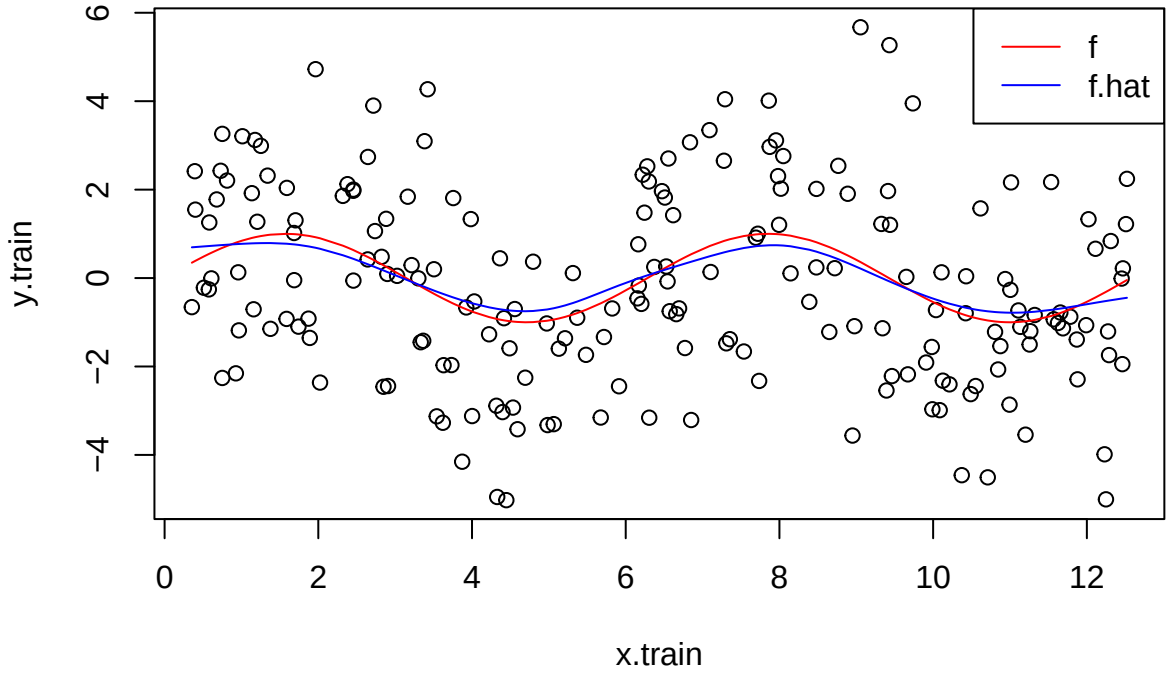
```
      W[i,j] = normal(x.train[i]-x.train[j],0,sigma[k])/sum(normal(x.train[i]-x.train,0,sigma[k]))

    }
  }
  variance[k] = noise.var * sum(diag(t(W)%*%W))/n
  bias[k] = squared_norm(W%*%f-f)/n
  summation[k] = variance[k] + bias[k] # MSE
}

# Plotting
plot(sigma, bias, type = 'l', col = 'red', xlab = 'sigma', ylab='', ylim=c(0, 2))
lines(sigma, variance, col = 'blue')
lines(sigma, summation, col = 'green')
legend('left', legend = c("Bias", "Variance", "Sum"),
       col = c("red", "blue", "green"), pch=16)
```



(b) Plot the training sample, f, and $\hat{f}$ for the optimal choice of σ.

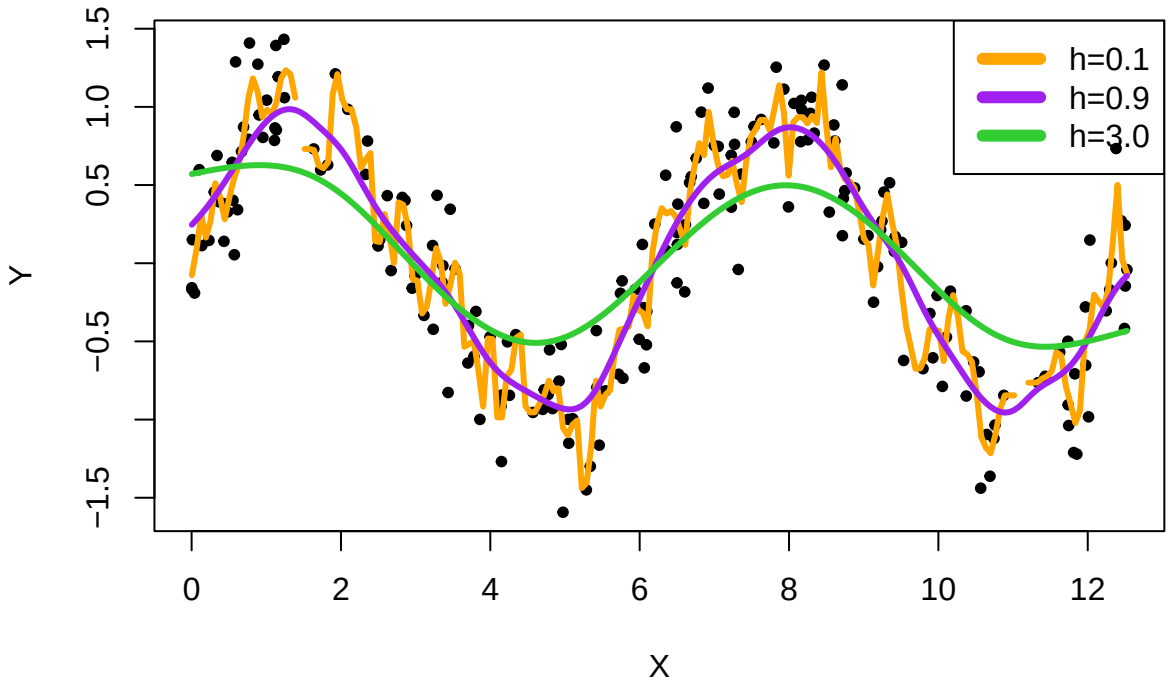
```
opt = sigma[which.min(summation)]
W = matrix(nrow = n, ncol = n)
for (i in 1:n) {
  for (j in 1:n) {

    W[i,j] = normal(x.train[i] - x.train[j], 0, opt)/sum(normal(x.train[i] - x.train, 0, opt))

  }
```

```
}
plot(x.train, y.train)
lines(x.train, f, col = 'red')
f.hat = W %*% f
lines(x.train, f.hat, col = 'blue')
legend('topright', legend = c("f", "f.hat"), col = c("red", "blue"), lty = 1)
```



(c) Check the output for simulated data in Part 1.

```
Kreg1 = ksmooth(x=X,y=Y,kernel = "normal",bandwidth = 0.1)
Kreg2 = ksmooth(x=X,y=Y,kernel = "normal",bandwidth = 0.9)
Kreg3 = ksmooth(x=X,y=Y,kernel = "normal",bandwidth = 3.0)
plot(X,Y,pch=20)
lines(Kreg1, lwd=3, col="orange")
lines(Kreg2, lwd=3, col="purple")
lines(Kreg3, lwd=3, col="limegreen")
legend("topright", c("h=0.1","h=0.9","h=3.0"), lwd=6,
col=c("orange","purple","limegreen"))
```


h=0.1
h=0.9
h=3.0
Y
X
1.5
1.0
0.5
−0.5
−1.5
0
2
4
6
8
10
12