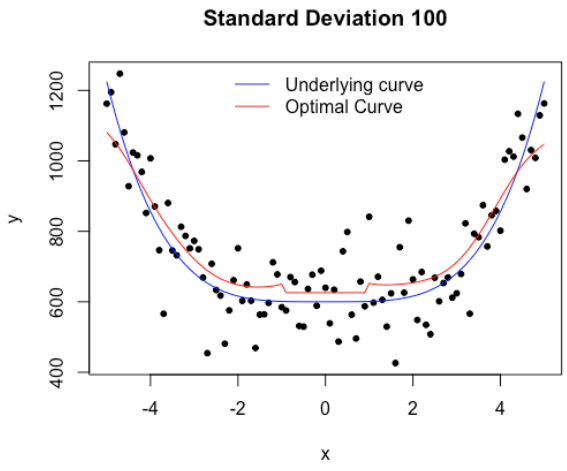
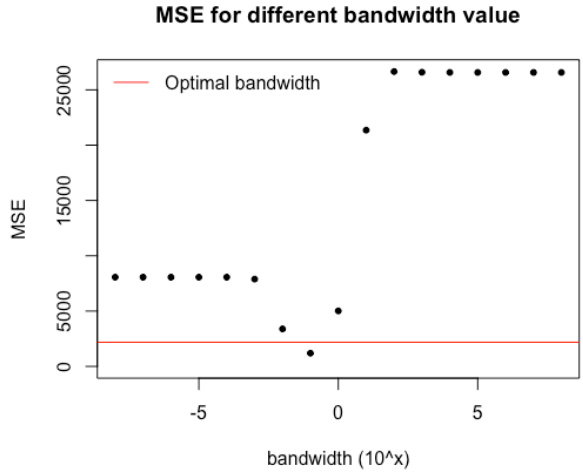
case 1 - globally solve:

```
gauss.kernel = function(x, y,ind,b){
  dv = x[ind]-x
  k = exp(-(dv^2)/(2*b))*(1/sqrt(2*pi*b))
  yhat = sum(k*y)/sum(k)
  return(yhat)
}

gaussian.smooth = function(x,y,b,u2){
  lb = 10
  ub = 10^5
  yhat = vector(length = length(y))

  for(i in 1:length(y)){

    win = c(x[i]-3*b[i], x[i]+3*b[i]) # window

    # take average in the window,
    # and set range of window is 1/10 of range of x
    mean.win = ifelse(range(win)[2]< range(x)[2] & range(win)[1]>
range(x)[1], mean(y[which(x>win[1] & x <win[2])]), mean(y[which(x>
range(x)[1]/10 & x< range(x)[2]/10 )]))

    yhat[i] = ifelse(u2[i]<lb, mean.win,
                     ifelse(u2[i]>ub, y[i], gauss.kernel(x,y,i,b[i])))
  }
```





case 2 - locally solve:

```
gauss.kernel = function(x, y,ind,b, win, max.range){
  if(b == 0){
    b = 0.00000000001
  }

  ran = (range(x)[2] - range(x)[1])/max.range
  xfix = x[ind]

  ub =  min(xfix + win*b, xfix + ran/2) #upperbound
  lb =  max(xfix - win*b, xfix - ran/2) #lowerbound

  xnew = x[x >= lb & x <= ub]
  ynew = y[x >= lb & x <= ub]

  k = exp(-((xfix - xnew)^2)/(2*b))
  yhat = sum(k*ynew)/sum(k)
  return(yhat)
}


gaussian.smooth = function(x,y,b){
  yhat = vector(length = length(y))
  for(i in 1:length(y)){
    yhat[i] = gauss.kernel(x,y,i,b[i], 10,10)  # adaptive bandwidth
  }
  return(yhat)
}
```