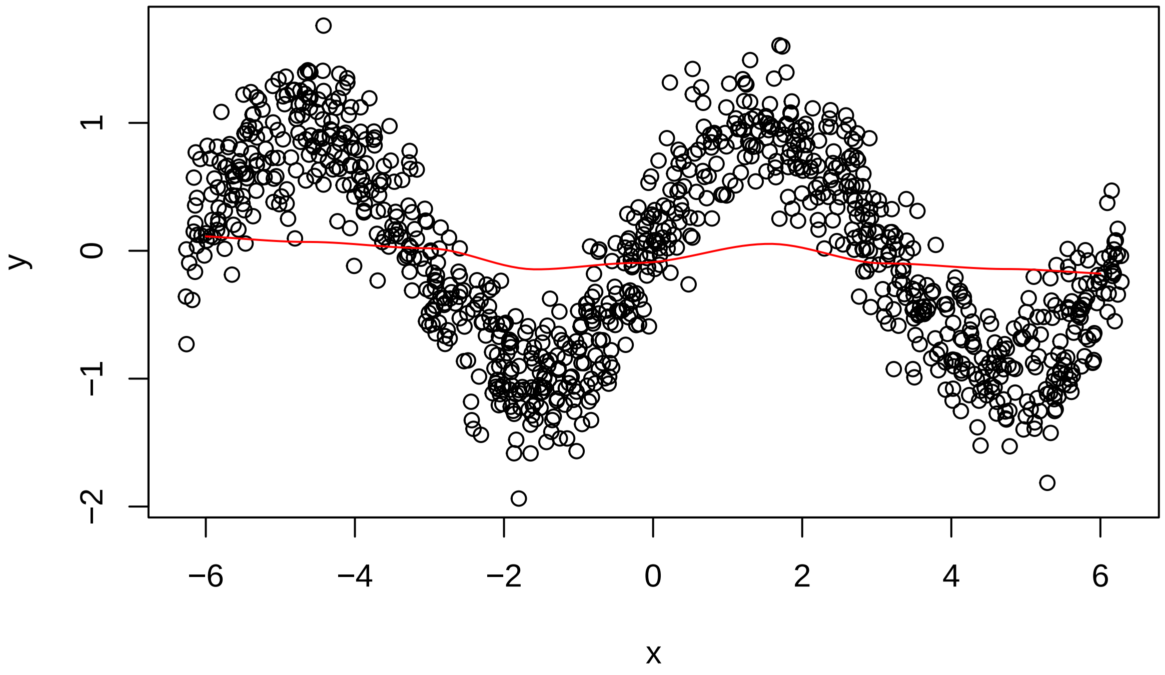
kernel regression

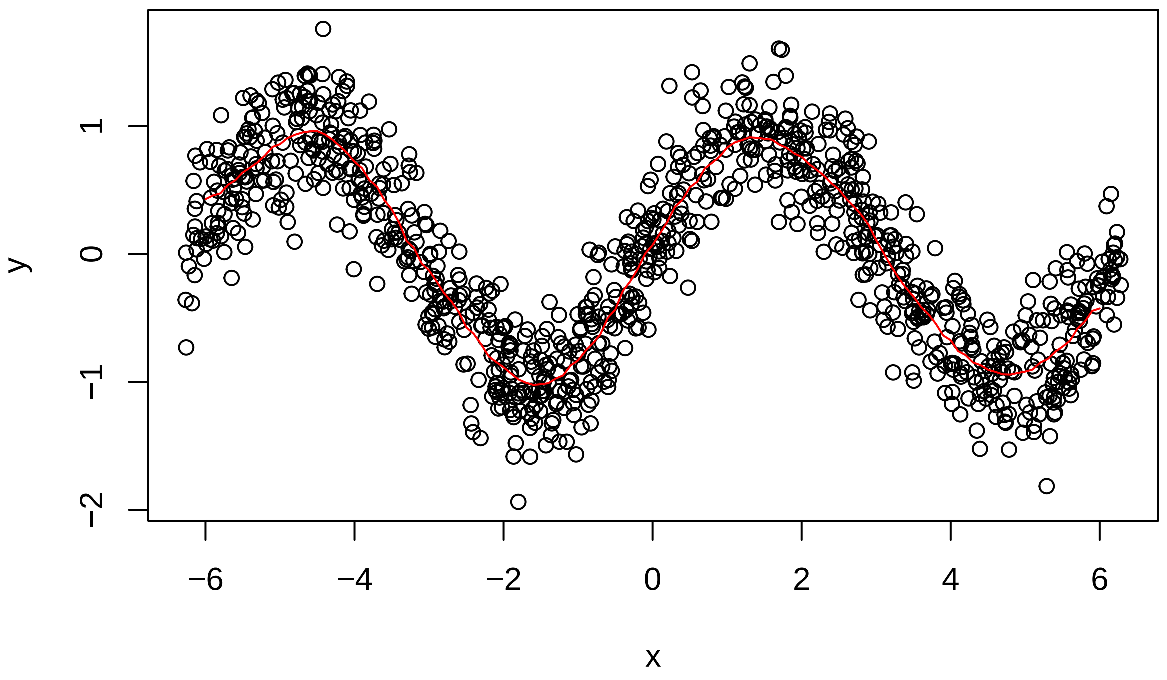
Yang Wang

2/9/2023

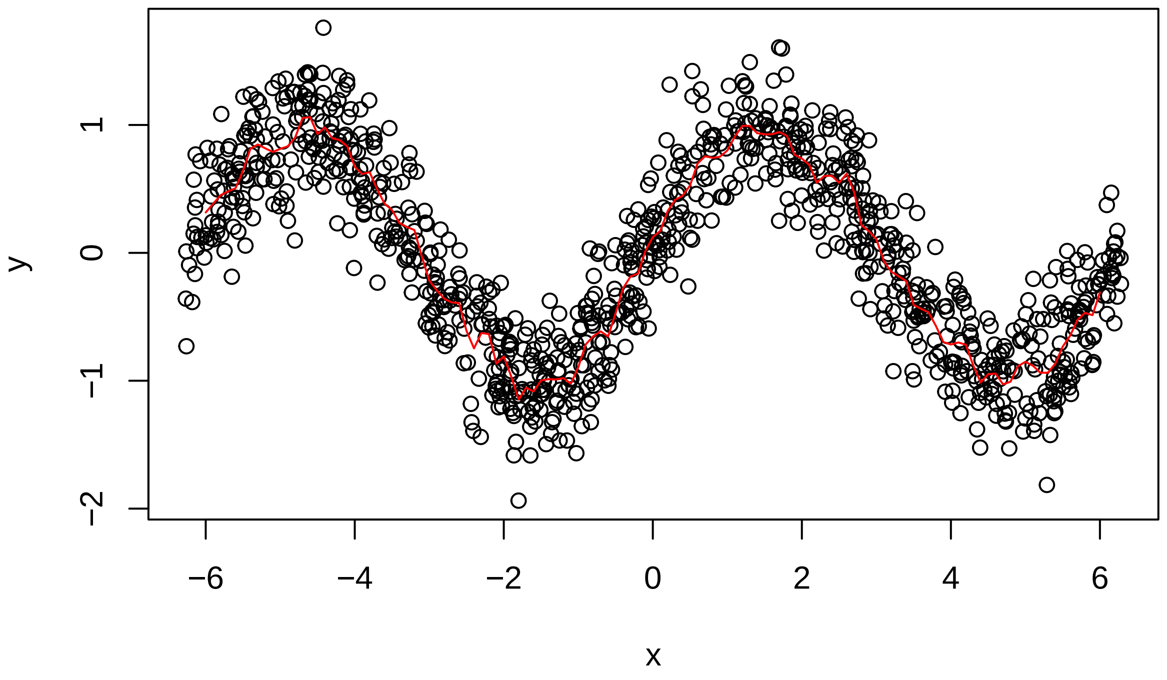
```r
# simulate a dataset
set.seed(1)
x = runif(1000, min = -2*pi, max = 2*pi)
y = sin(x) + rnorm(1000, 0, 0.3)


# nw kernel regression
lo.mod = loess(y ~ x, degree = 0 )
x.plot = seq(from =-6, to = 6, by = 0.1)
y.fit = predict(lo.mod, newdata = x.plot)

#plot the fit and observations
plot(x,y)
lines(y.fit~x.plot, type = "l", col = "red")
```



```r
# decrease the bandwidth
lo.mod = loess(y ~ x, degree = 0, span = 0.1 )
y.fit = predict(lo.mod, newdata = x.plot)

plot(x,y)
lines(y.fit~x.plot, type = "l",col = "red")
```



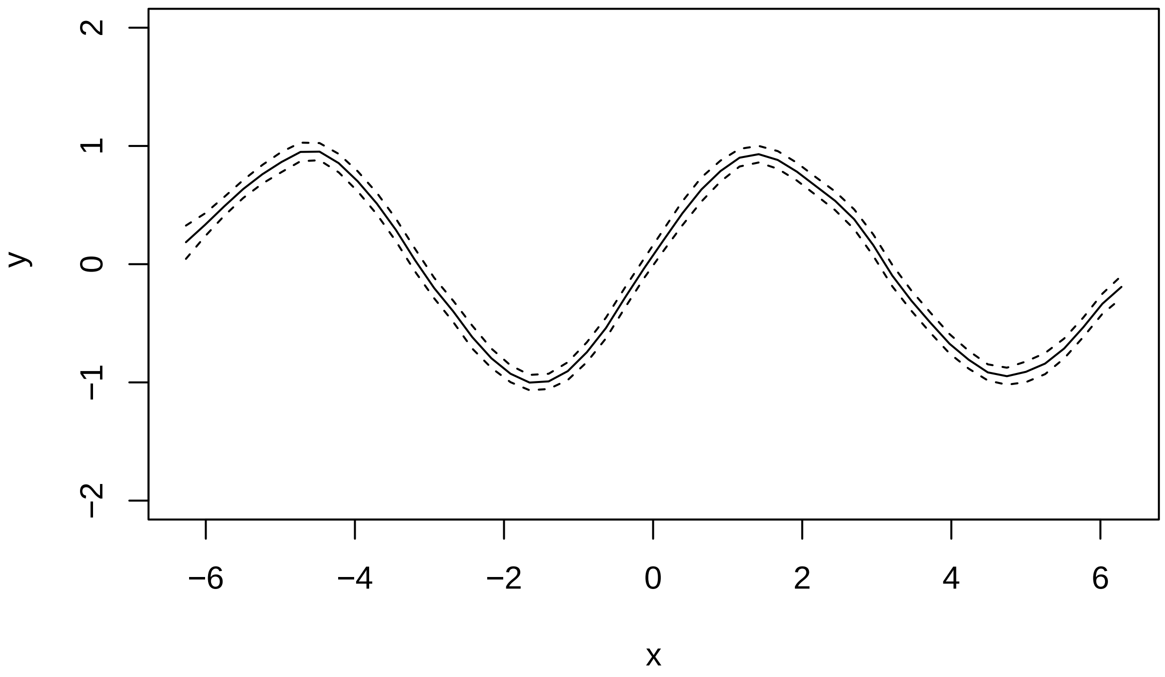
```r
# decrease the band width more
lo.mod = loess(y ~ x, degree = 0, span = 0.025 )
y.fit = predict(lo.mod, newdata = x.plot)

plot(x,y)
lines(y.fit~x.plot, type = "l",col = "red")
```



```r
#construct confidence intervals for NW estimator, use h = 0.25
library(np)
```

```
## Nonparametric Kernel Methods for Mixed Datatypes (version 0.60-16)
## [vignette("np_faq",package="np") provides answers to frequently asked questions]
## [vignette("np",package="np") an overview]
## [vignette("entropy_np",package="np") an overview of entropy-based methods]
```

```
bw = npregbw(formula= y~x, bws = 0.25, bwtype ="fixed", bandwidth.compute = F)
model = npreg(bws = bw, gradients= TRUE)
plot(model, plot.errors.method = "asymptotic", plot.errors.style = "band", ylim = c(-2,2))
```

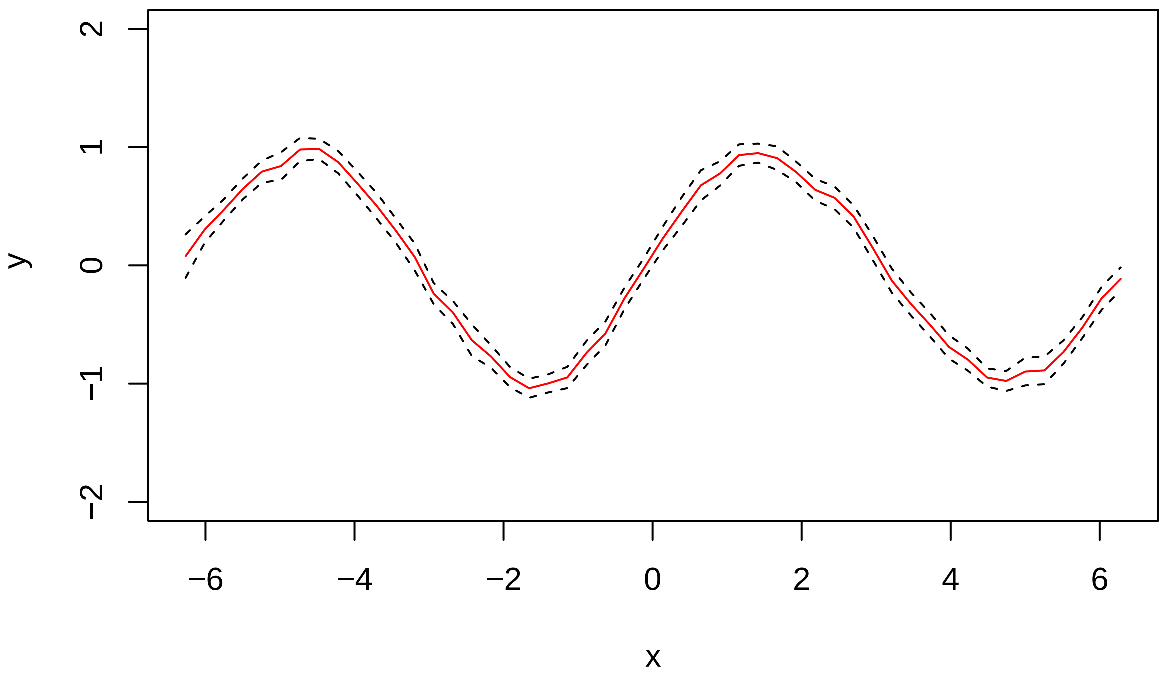


```
#points(y~x, col = "gray")
```

```
#construct confidence intervals for NW estimator, use optimal H computed by the function
library(np)
bw = npregbw(formula= y~x,bwtype ="fixed", bandwidth.compute = T)
```

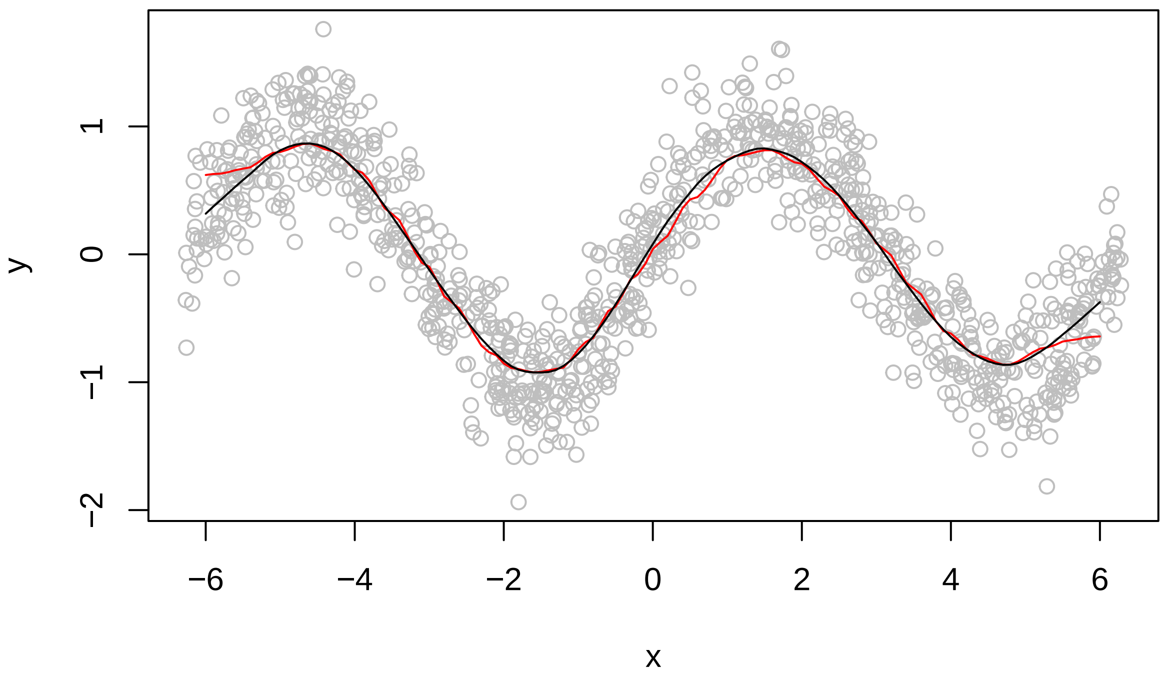
```
## Multistart 1 of 1 |Multistart 1 of 1 |Multistart 1 of 1 |Multistart 1 of 1 /Multistart 1 of 1 |Multis
```

```
model = npreg(bws = bw, gradients= TRUE)
plot(model, plot.errors.method = "asymptotic", plot.errors.style = "band", ylim = c(-2,2), col = "red")
```

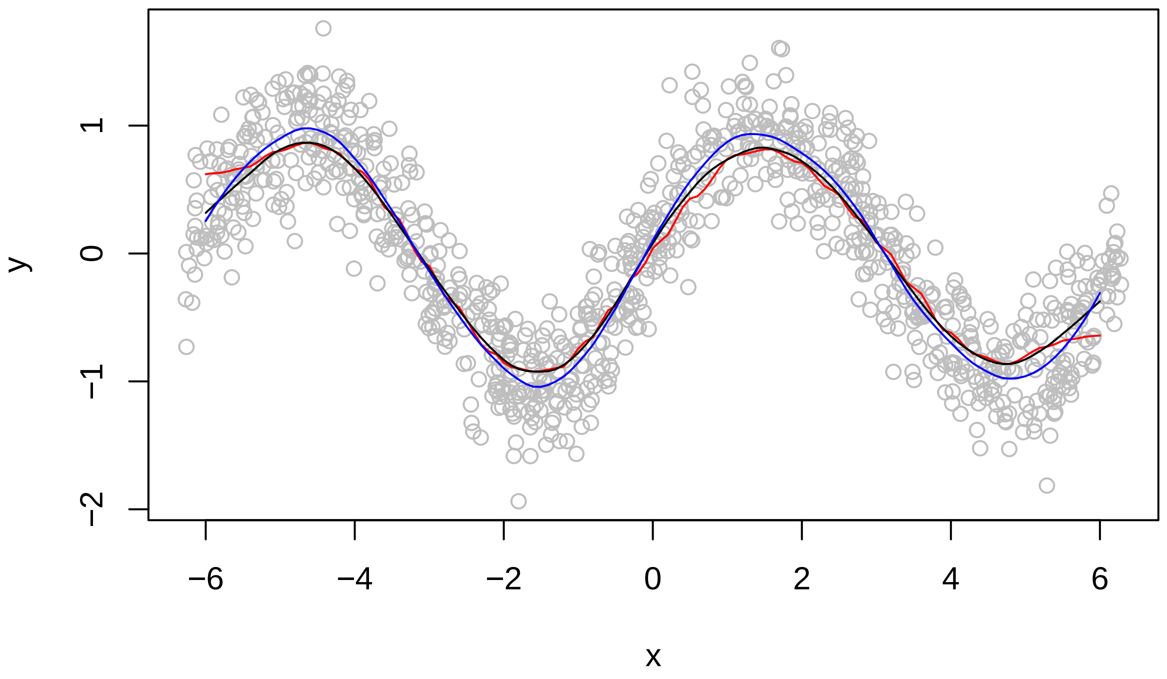


```
#points(y~x, col = "gray")
```

```
# fit a local linear regression model
# and compare local linear and local weighted regression models
lo.linear = loess(y ~ x, degree = 1, span = 0.2 )
lo.mod = loess(y~x, degree = 0, span = 0.2)

y.fit.linear = predict(lo.linear, newdata = x.plot)
y.fit.mod = predict(lo.mod, newdata = x.plot)

plot(x, y, col = "gray")
points(y.fit.mod~x.plot, type = "l",col = "red")
lines(y.fit.linear~x.plot, type = "l",col = "black")
```



```r
# fit a local linear regression model
# and compare local quadratic and local weighted regression models
lo.quadratic = loess(y ~ x, degree = 2, span = 0.2 )
y.fit.quadratic = predict(lo.quadratic, newdata = x.plot)

plot(x, y, col = "gray")
lines(y.fit.mod~x.plot, type = "l",col = "red")
lines(y.fit.linear~x.plot, type = "l",col = "black")
lines(y.fit.quadratic~x.plot, type = "l",col = "blue")
```



```
# multidimensional kernel regression
# simulate the data
library(mvtnorm)
set.seed(111)
x = runif(1000, min = -2.1, max = 2.1)
z = runif(1000, min = -2.1, max = 2.1)
y = dmvnorm(cbind(x, z), sigma = diag(2)*0.3) + rnorm(1000, mean=0,sd=1)
```

```
# fit a two dimensional data
lo.mod = loess(y ~ x+z, degree = 1, span = 0.2)
x = seq(from = -2, to=2, by = 0.1)
z = seq(from = -2, to=2, by = 0.1)
dat = expand.grid(x,z)
y.fit = predict(lo.mod, newdata = as.matrix(dat))
y.fit = matrix(data = y.fit, nrow = length(x), ncol =length(z) )
# plot the fit
persp(x,z,y.fit, phi = 30, theta = 160)
```