4850_assignment1

Yufei xia

2020/2/24

```
theta1 = numeric(0)
theta2 = numeric(0)

simulate = function(sigma_sq){
beta0 = 1
beta1 = 1
x =rnorm(1000,4,1)
y = beta0 + beta1*x + rnorm(1000,0,sqrt(sigma_sq))
xstar = x + rnorm(1000,0,sigma_sq)
model2 = lm(y~xstar)
model1 = lm(y~x)
return(c(summary(model1)$coefficients[2,1]-1,summary(model2)$coefficients[2,1]-1))
}

sigma = c(0.15,0.55,0.75)

sigma1= replicate(n = 1000, simulate(0.15))
print("the bias for beta1 and beta2 when varience = 0.15")
```

```
## [1] "the bias for beta1 and beta2 when varience = 0.15"
```

```
apply(sigma1,1,mean)
```

```
## [1] -0.0001667277 -0.0221529250
```

```
print("the varience for beta1 and beta2 when varience = 0.15")
```

```
## [1] "the varience for beta1 and beta2 when varience = 0.15"
```

```
apply(sigma1,1,var)
```

```
## [1] 0.0001475472 0.0001630140
```

```
sigma1= replicate(n = 1000, simulate(0.55))
print("the bias for beta1 and beta2 when varience = 0.55")
```

```
## [1] "the bias for beta1 and beta2 when varience = 0.55"
```

```
apply(sigma1,1,mean)
```

```
## [1]  3.205932e-06 -2.332582e-01
```

```
print("the varience for beta1 and beta2 when varience = 0.55")
```

```
## [1] "the varience for beta1 and beta2 when varience = 0.55"
```

```
apply(sigma1,1,var)
```

```
## [1] 0.0004975920 0.0005924441
```

```
sigma1= replicate(n = 1000, simulate(0.75))
print("the bias for beta1 and beta2 when varience = 0.75")
```

```
## [1] "the bias for beta1 and beta2 when varience = 0.75"
```

```
apply(sigma1,1,mean)
```

```
## [1]  0.000146004 -0.358906868
```

```
print("the varience for beta1 and beta2 when varience = 0.75")
```

```
## [1] "the varience for beta1 and beta2 when varience = 0.75"
```

```
apply(sigma1,1,var)
```

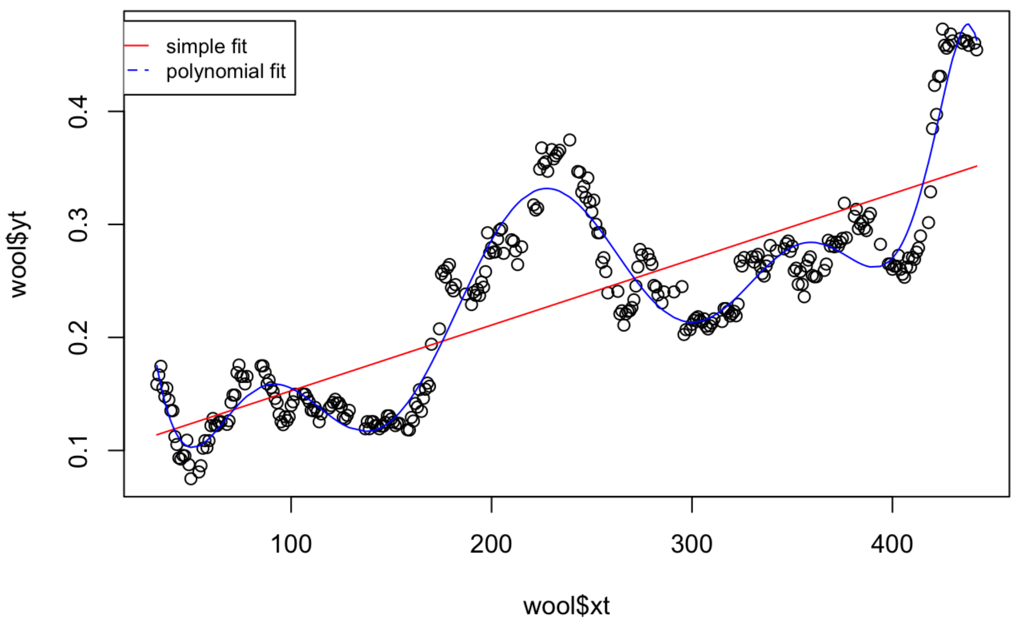
```
## [1] 0.0007707348 0.0007235687
```

```
#result = apply(m,1,mean)
#print(result)
```

from the result we can see the bias for beta1 doesn't grow as the sigma increase, however, the bias for second beta become bigger as the sigma go up, which prove our part b, w2 is not equal to 1 and decrease as sigma increase. in term of the varience of beta1 and beta2, we know that they are increase as sigma go up. which satisfy with the formula we calculated in the partc.also the varience is equal according to question 3.

question4 part1

```
wool = read.csv("wool.txt",sep = "")
model_fit = lm("yt~xt",data = wool)
model_fit2 = lm("yt~poly(xt,10)",data = wool)
plot(wool$xt,wool$yt)
lines(wool$xt,predict(model_fit),col = "red")
lines(wool$xt,predict(model_fit2),col = "blue")
legend(1,0.48,legend=c("simple fit", "polynomial fit"),col=c("red", "blue"),lty=1:2, cex=0.8)
```



part2

```r
llg_normal = function(data,x,h){
  train_x = as.numeric(data[,1])
  train_y = as.numeric(data[,2])
  vx = train_x-x
  w = as.numeric(dnorm(vx,0,h))
  weight = w/sum(w)
  oldw =getOption("warn")
  options(warn = -1)
  fit = lm(train_y ~ vx,weights = weight)
  options(warn = oldw)
  response = fit$coef[1]
  as.numeric(response)
  }


leave_out_normal = function(i,data,h){
  xi = data[i,1]
  newdata = data[-i,]
  yi = data[i,2]
  response = llg_normal(newdata,xi,h)
  c(yi,response)
}

cv.glm_normal  =function(data,h){
  index = as.matrix(1:nrow(data))
  output = apply(index,1,leave_out_normal,data =data,h=h)
  output<-t(output)
  error = sum((output[,1]-output[,2])^2)
  error
  }


cvglm_linear_normal = function(data,interval){
  optimize(cv.glm_normal,interval = interval,data = data)$minimum
}

optmized_bindwith_linear = cvglm_linear_normal(wool,c(0,length(wool[,1])))
print("the optimized bindwith is for linear fit is")
```

```
## [1] "the optimized bindwith is for linear fit is"
```

```r
optmized_bindwith_linear
```

```
## [1] 1.74673
```

```r
response_linear= NULL

for (i in 1:length(wool[,1]) ){
  response_linear[i] = llg_normal(wool,wool[i,1],optmized_bindwith_linear)
}


linear_data = cbind(wool[,1],response_linear)
linear_data = linear_data[order(linear_data[,1]),]
```

```r
llg_normal_const = function(data,x,h){
  train_x = as.numeric(data[,1])
  train_y = as.numeric(data[,2])
  vx = train_x-x
  w = as.numeric(dnorm(vx,0,h))
  weight = w/sum(w)
  oldw =getOption("warn")
  options(warn = -1)
  fit = lm(train_y ~ 1,weights = weight)
  options(warn = oldw)
  response = fit$coef[1]
  as.numeric(response)
  }


leave_out_normal_const = function(i,data,h){
  xi = data[i,1]
  newdata = data[-i,]
  yi = data[i,2]
  response = llg_normal_const(newdata,xi,h)
  c(yi,response)
}
cv.glm_normal_const =function(data,h){
  index = as.matrix(1:nrow(data))
  output = apply(index,1,leave_out_normal_const,data =data,h=h)
  output<-t(output)
  error = sum((output[,1]-output[,2])^2)
  error
  }
cvglm_linear_normal_const = function(data,interval){
  optimize(cv.glm_normal_const,interval = interval,data = data)$minimum
}
optmized_bindwith_const = cvglm_linear_normal_const(wool,c(0,length(wool[,1])))
print("the optimized bindwith is for constant fit is ")
```

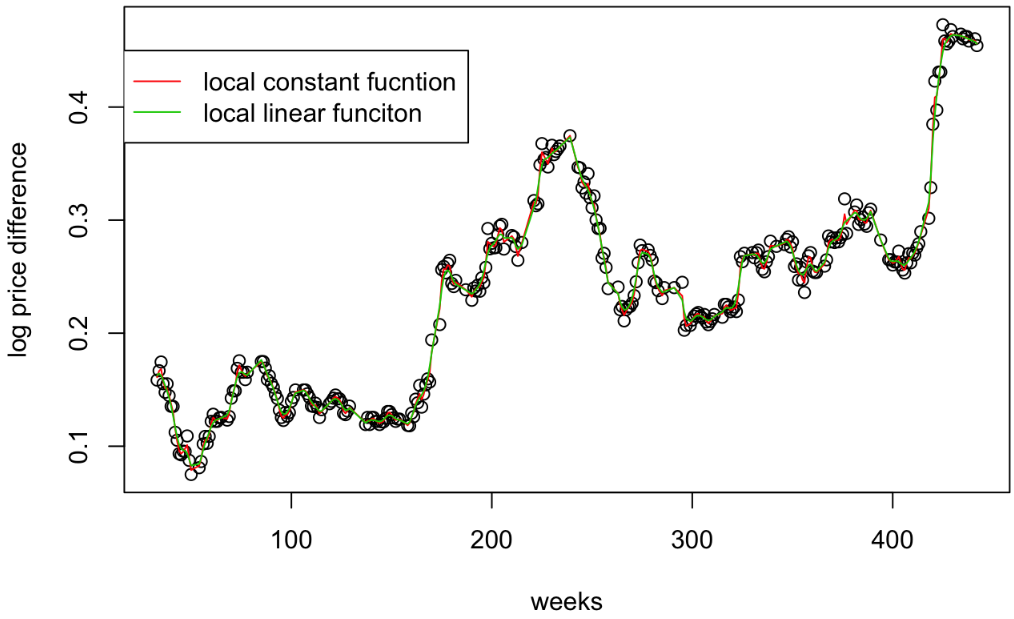
```
## [1] "the optimized bindwith is for constant fit is "
```

```r
optmized_bindwith_const
```

```
## [1] 0.7162969
```

```r
response_const= NULL
for (i in 1:length(wool[,1]) ){
  response_const[i] = llg_normal_const(wool,wool[i,1],optmized_bindwith_const)
}

const_data = cbind(wool[,1],response_const)
const_data = const_data[order(const_data[,1]),]
```

```r
plot(x = wool[,1],y=wool[,2],xlab="weeks",ylab ="log price difference")
lines(const_data[,1],const_data[,2],col="2")
lines(linear_data[,1],linear_data[,2],col="3")
legend(10,0.45,c("local constant fucntion","local linear funciton"),col=c(2,3),lty =c(1,1))
```

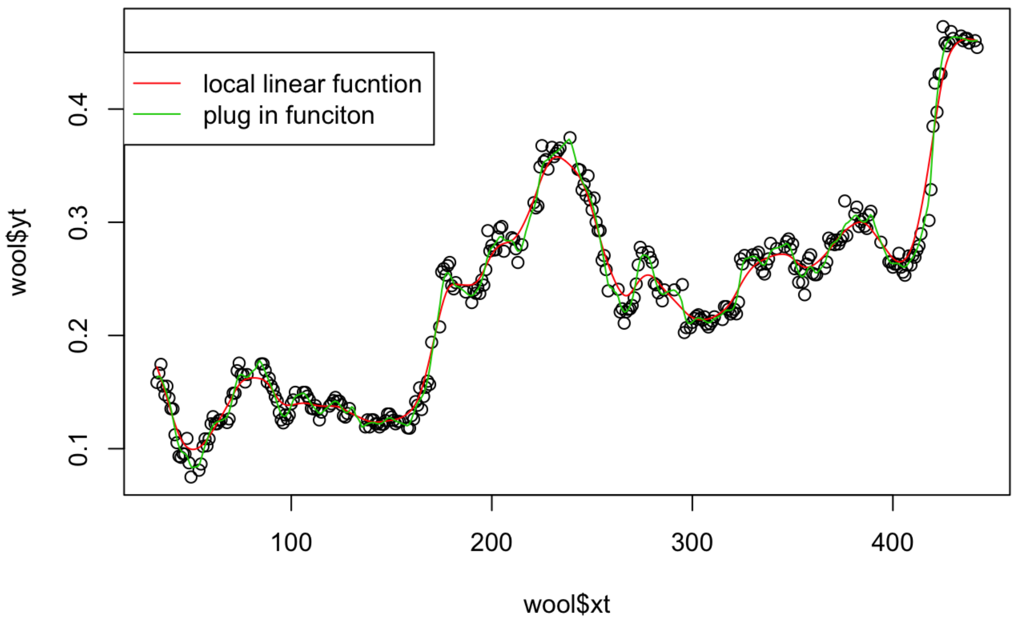


part c

```
library(KernSmooth)
```

```
## KernSmooth 2.23 loaded
## Copyright M. P. Wand 1997-2009
```

```
plot(wool$xt, wool$yt)
h <- dpill(wool$xt, wool$yt)
plugin_fit <- locpoly(wool$xt, wool$yt, bandwidth = h,degree = 1)
lines(plugin_fit,col="2")
linear_fit<- locpoly(wool$xt, wool$yt, bandwidth = optmized_bindwith_linear,degree = 1)
lines(linear_fit,col="3")
legend(10,0.45,c("local linear fucntion","plug in funciton"),col=c(2,3),lty =c(1,1))
```



part d

```
predy =list(predict(model_fit),predict(model_fit2),response_const,response_linear,plugin_fit$y)
sapply(predy,function(x){sum((x-wool[,2])^2)})
```

```
## Warning in x - wool[, 2]: longer object length is not a multiple of shorter
## object length
```

```
## [1] 0.962803050 0.160136905 0.004264583 0.012269386 5.622580024
```

we can see that from the result we get, the local constant estimator sum of square error is 0.0042645,which is lowest. and also the plug in method has largest SSE.also as we predicted,the more degree you have, the less SSE for training data,we polynomial fit has less SSE. also we can see that the local constant estimator has less SSE than local linear estimator.

qustion 5 part a

```
ky = read.csv("kyphosis.txt",sep ="")
newda = cbind(ky["age"],ifelse(ky["kyphosis"]=="absent",0,1))
library("KernSmooth")
library("locpol")
llg = function(data,x,h){
  train_x = as.numeric(data[,1])
  train_y = as.numeric(data[,2])
  vx = train_x-x
  w = as.numeric(dnorm(vx,0,h))
  weight = w/sum(w)
  oldw =getOption("warn")
  options(warn = -1)
  fit = glm(train_y ~ vx,family = binomial(link=logit),weights = weight)
  options(warn = oldw)
  beta0 = fit$coef[1]
  p = exp(beta0)/(1+exp(beta0))
  as.numeric(p)
  }


leave_out = function(i,data,h){
  xi = data[i,1]
  newdata = data[-i,]
  yi = data[i,2]
  fit = llg(newdata,xi,h)
  c(yi,fit)
}

cv.glm  =function(data,h){
  index = as.matrix(1:nrow(data))
  output = apply(index,1,leave_out,data =data,h=h)
  output<-t(output)
  error = -sum(output[,1]*log(output[,2])+(1-output[,1])*log(1-output[,2]))
  error
  }


cvglm_linear = function(data,interval){
  optimize(cv.glm,interval = interval,data = data)$minimum
}


bindwith_linear = cvglm_linear(newda,c(0,81))
print("the bindwith using local linear function is")
```

```
## [1] "the bindwith using local linear function is"
```

```
bindwith_linear
```

```
## [1] 43.26988
```

```r
prob_est_linear = NULL

for (i in 1:length(newda[,1]) ){
  prob_est_linear[i] = llg(newda,newda[i,1],bindwith_linear)
}
print("the estimated pi is equal to")
```

```
## [1] "the estimated pi is equal to"
```

```r
prob_est_linear
```

```
##  [1] 0.24310999 0.17672473 0.27561089 0.04573366 0.04382350 0.04382350
##  [7] 0.21587319 0.14043253 0.29335165 0.20999094 0.26769889 0.21748010
## [13] 0.08318262 0.04382350 0.13317788 0.04382350 0.25950847 0.10384148
## [19] 0.26371814 0.10910286 0.09435296 0.29406893 0.28838169 0.26935129
## [25] 0.07523235 0.06056700 0.05829360 0.29171142 0.04971190 0.20582989
## [31] 0.12140588 0.28095547 0.27153938 0.29375801 0.24518500 0.28508882
## [37] 0.04382350 0.18855913 0.08869083 0.28253149 0.24804525 0.13403371
## [43] 0.23542569 0.21587319 0.28933000 0.24825660 0.25690337 0.26935129
## [49] 0.28666544 0.09592561 0.23536369 0.06056700 0.24825660 0.04573366
## [55] 0.24518500 0.24560137 0.04573366 0.28784266 0.18541169 0.29290165
## [61] 0.27153938 0.29285186 0.26573772 0.28990076 0.28990076 0.08048966
## [67] 0.03952567 0.17241275 0.08318262 0.07523235 0.17672473 0.27749407
## [73] 0.27658826 0.01966326 0.06526337 0.09206763 0.18100784 0.10609131
## [79] 0.28784266 0.15655665 0.13722761
```

```r
cvglm_const = function(data,interval){
optimize(cv.glm_const,interval = interval,data = data)$minimum
}

 lc = function(data,x,h){
  train_x = as.numeric(data[,1])
  train_y = as.numeric(data[,2])
  vx = train_x-x
  w = as.numeric(dnorm(vx,0,h))
  weight = w/sum(w)
  oldw =getOption("warn")
  options(warn = -1)
  fit = glm(train_y ~ 1,family = binomial(link=logit),weights = weight)
  options(warn = oldw)
  beta0 = fit$coef[1]
  p = exp(beta0)/(1+exp(beta0))
  as.numeric(p)
  }


leave_out_const = function(i,data,h){
  xi = data[i,1]
  newdata = data[-i,]
  yi = data[i,2]
  fit = lc(newdata,xi,h)
  c(yi,fit)
}

cv.glm_const  =function(data,h){
  index = as.matrix(1:nrow(data))
  output = apply(index,1,leave_out_const,data =data,h=h)
  output<-t(output)
  error = -sum(output[,1]*log(output[,2])+(1-output[,1])*log(1-output[,2]))
  error
  }


bindwith_const = cvglm_const(newda,c(0,81))
print("the optimized bindwith for local regression is")
```

```
## [1] "the optimized bindwith for local regression is"
```

```
bindwith_const
```

```
## [1] 29.02671
```

```
prob_est_const = NULL
print("the estimated pi equal to")
```

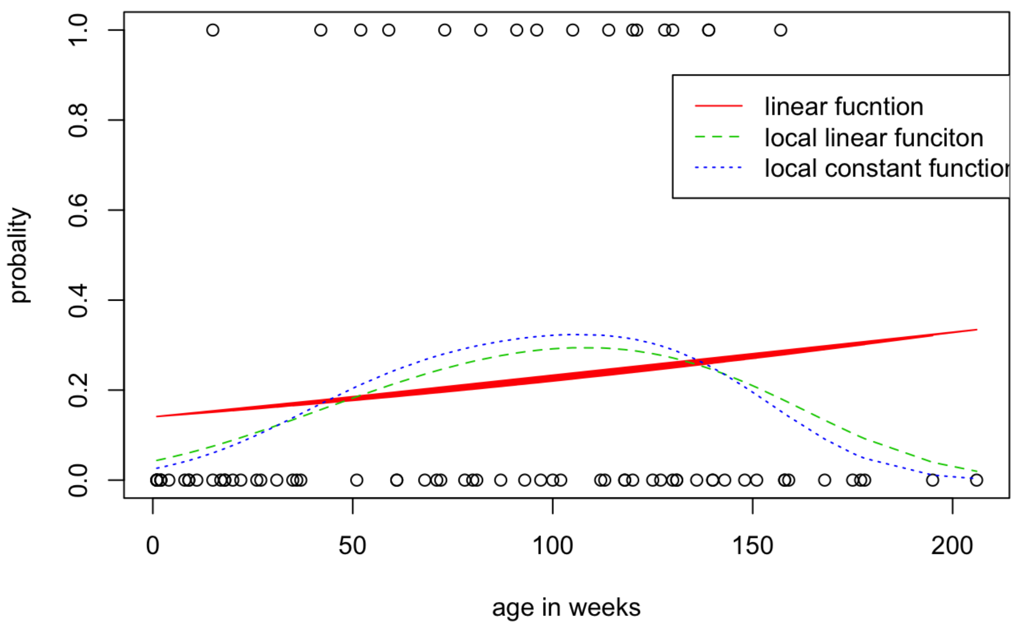
```
## [1] "the estimated pi equal to"
```

```
for (i in 1:length(newda[,1]) ){
  prob_est_const[i] = llg(newda,newda[i,1],bindwith_const)
}

prob_est_const
```

```
##  [1] 0.275743625 0.147288812 0.295750239 0.028287096 0.026480290
##  [6] 0.026480290 0.245624804 0.147329752 0.321507174 0.238681282
## [11] 0.300108093 0.206897204 0.070126783 0.026480290 0.091267560
## [16] 0.026480290 0.292270121 0.059567199 0.296330818 0.104068077
## [21] 0.084409586 0.323550673 0.318900913 0.286403458 0.060377246
## [26] 0.043519368 0.041056009 0.321760437 0.032169313 0.189350188
## [31] 0.120924001 0.303609668 0.289686231 0.322138885 0.249453386
## [36] 0.315997267 0.026480290 0.212155652 0.077092531 0.313716307
## [41] 0.280840718 0.138437001 0.234394730 0.245624804 0.319726957
## [46] 0.254190822 0.267486291 0.286403458 0.311866917 0.051965782
## [51] 0.267521670 0.043519368 0.254190822 0.028287096 0.249453386
## [56] 0.278330299 0.028287096 0.313552973 0.208110025 0.322727533
## [61] 0.289686231 0.320756948 0.298254536 0.316494366 0.316494366
## [66] 0.066781827 0.011132192 0.141330884 0.070126783 0.060377246
## [71] 0.147288812 0.298533375 0.308344056 0.003206261 0.048742208
## [76] 0.048416120 0.153284160 0.099995704 0.313552973 0.169599816
## [81] 0.142876721
```

part2

```
#assume it is linear function bew
#so we have
fit = glm(newda[,2]~newda[,1],family = binomial)
plot(newda[,1],newda[,2],xlab ="age in weeks", ylab ="probality")
prob = function(t){
  exp(fit$coef[1]+fit$coef[2]*t)/(1+exp(fit$coef[1]+fit$coef[2]*t))
}


model_linear = cbind(newda[,1],prob_est_linear)
model_linear = model_linear[order(model_linear[,1]),]
model_const = cbind(newda[,1],prob_est_const)
model_const = model_const[order(model_const[,1]),]
lines(newda[,1],prob(newda[,1]),col ="2",lty =1)
lines(model_linear[,1],model_linear[,2],col ="3",lty =2 )
lines(model_const[,1],model_const[,2],col="4",lty =3)
legend(130,0.9,c("linear fucntion","local linear funciton","local constant function"),col=c(2,3,4),lty =c(1,
2,3))
```




```
#we can get that the plot is not likely to be linear
```

from the plot we get, apprantly the linear assumption for f(x) is not reasonable beacause the plot doesn’t follow the same pattern for linear function and local function.

question 6 part b

```
crab = read.csv("crab.txt",sep ="")
llg_pos = function(data,x,h){
  train_x = as.numeric(data[,1])
  train_y = as.numeric(data[,2])
  vx = train_x-x
  w = as.numeric(dnorm(vx,0,h))
  weight = w/sum(w)
  oldw =getOption("warn")
  options(warn = -1)
  fit = glm(train_y ~ vx,family = poisson(link = "log"),weights = weight)
  options(warn = oldw)
  sigma = exp(fit$coef[1])
  as.numeric(sigma)
  }


leave_out_pos = function(i,data,h){
  xi = data[i,1]
  newdata = data[-i,]
  yi = data[i,2]
  sigma = llg_pos(newdata,xi,h)
  c(yi,sigma)
}

cv.glm_pos  =function(data,h){
  index = as.matrix(1:nrow(data))
  output = apply(index,1,leave_out_pos,data =data,h=h)
  output<-t(output)
  error = -sum(output[,1]*log(output[,2])-output[,2])
  error
  }

cvglm_linear_pos = function(data,interval){
  optimize(cv.glm_pos,interval = interval,data = data)$minimum
}
pos_bindwith = cvglm_linear_pos(crab,interval = c(0,length(crab[,1])))
print("the optimized bw is")
```

```
## [1] "the optimized bw is"
```

```
pos_bindwith
```

```
## [1] 3.434197
```

```
POS_LINEAR =NULL
for(i in 1:length(crab[,1])){
POS_LINEAR[i] = llg_pos(crab,crab[i,1],pos_bindwith)
}
print("the predicted mu is")
```

```
## [1] "the predicted mu is"
```

```
POS_LINEAR
```

```
##   [1] 3.8842716 2.6439340 2.4461765 0.7720491 4.2726315 2.1637376 2.7454330
##   [8] 2.1184143 2.4949353 3.4395078 2.6944741 4.2175941 4.9568645 1.3243807
##  [15] 2.7454330 1.9423195 4.8056674 2.7454330 2.3500736 2.3500736 3.4395078
##  [22] 3.1665786 1.7341816 4.1069586 2.9006976 1.9423195 3.3296433 2.9006976
##  [29] 2.1637376 1.0385643 4.9070163 2.3500736 2.1184143 2.5441542 3.2750370
##  [36] 5.0547977 2.1637376 4.8056674 1.3243807 1.6941797 2.6439340 2.5441542
##  [43] 4.2726315 2.9006976 1.1918846 1.6547244 1.8574480 2.6439340 2.0293183
##  [50] 1.1918846 4.1069586 4.4362640 3.0060608 1.5023936 3.5500949 3.8284733
##  [57] 2.0293183 2.4949353 3.6056017 3.1665786 4.2726315 2.4461765 1.8158178
##  [64] 2.4949353 1.3939310 3.9957733 4.6498802 1.3939310 1.9423195 3.4395078
##  [71] 2.7967985 3.6056017 5.6655425 2.1637376 2.7454330 3.9400476 1.9423195
##  [78] 3.6612185 2.1637376 4.2726315 5.5866757 3.4947224 1.9423195 1.6547244
##  [85] 3.8284733 1.7747283 3.7169199 2.6439340 2.0293183 2.5441542 3.2206755
##  [92] 3.3844741 3.0060608 3.0592535 2.5441542 1.6158170 3.6612185 4.8056674
##  [99] 2.1637376 3.5500949 3.8842716 2.3978865 2.6439340 2.7454330 1.3588802
## [106] 1.3243807 2.2095711 2.5938237 2.3978865 3.0592535 4.2726315 3.9957733
## [113] 2.0293183 4.2726315 3.1665786 1.6158170 3.1665786 1.8158178 1.1918846
## [120] 2.2095711 2.1184143 3.4395078 1.8574480 4.5438212 2.7454330 2.0293183
## [127] 4.7022767 2.4949353 2.7454330 3.1665786 2.0736063 1.6158170 3.8284733
## [134] 2.2559091 1.4295335 2.5441542 3.4395078 2.4949353 3.0592535 3.4395078
## [141] 3.9957733 3.9957733 3.3844741 3.2750370 3.2206755 3.7169199 2.9006976
## [148] 1.3588802 2.6439340 1.9423195 2.5441542 1.5396508 3.0060608 2.3978865
## [155] 3.8284733 2.2559091 2.3027454 2.4949353 4.4362640 1.6547244 3.3844741
## [162] 2.7454330 3.7169199 3.9400476 6.2138788 2.5441542 1.7341816 1.3939310
## [169] 3.8842716 2.9006976 2.9006976 2.6944741 1.9423195
```

```
pos_linear_data = cbind(crab$x,POS_LINEAR)
pos_linear_data = pos_linear_data[order(pos_linear_data[,1]),]
```

```r
llg_pos_const = function(data,x,h){
  train_x = as.numeric(data[,1])
  train_y = as.numeric(data[,2])
  vx = train_x-x
  w = as.numeric(dnorm(vx,0,h))
  weight = w/sum(w)
  oldw =getOption("warn")
  options(warn = -1)
  fit = glm(train_y ~ 1,family = poisson(link = "log"),weights = weight)
  options(warn = oldw)
  sigma = exp(fit$coef[1])
  as.numeric(sigma)
  }


leave_out_pos_const = function(i,data,h){
  xi = data[i,1]
  newdata = data[-i,]
  yi = data[i,2]
  sigma = llg_pos_const(newdata,xi,h)
  c(yi,sigma)
}

cv.glm_pos_const  =function(data,h){
  index = as.matrix(1:nrow(data))
  output = apply(index,1,leave_out_pos_const,data =data,h=h)
  output<-t(output)
  error = -sum(output[,1]*log(output[,2])-output[,2])
  error
  }

cvglm_linear_pos_const = function(data,interval){
  optimize(cv.glm_pos_const,interval = interval,data = data)$minimum
}
print("the optimized bw is")
```

```
## [1] "the optimized bw is"
```

```r
pos_bindwith_const = cvglm_linear_pos_const(crab,interval = c(0,length(crab[,1])))
pos_bindwith_const
```

```
## [1] 1.206288
```

```r
print("the predicted mu is")
```

```
## [1] "the predicted mu is"
```

```r
POS_CONST =NULL
for(i in 1:length(crab[,1])){
POS_CONST[i] = llg_pos_const(crab,crab[i,1],pos_bindwith_const)
}
POS_CONST
```

```
##   [1] 3.8423633 2.7768852 2.6397275 0.9223349 4.1405158 2.4307243 2.8491399
##   [8] 2.3935108 2.6735550 3.4370666 2.8125607 4.1026216 4.4463634 1.5241577
##  [15] 2.8491399 2.2347067 4.4128154 2.8491399 2.5718768 2.5718768 3.4370666
##  [22] 3.1869458 2.0151431 4.0218900 2.9656929 2.2347067 3.3346342 2.9656929
##  [29] 2.4307243 1.2099792 4.4382596 2.5718768 2.3935108 2.7075649 3.2844260
##  [36] 4.4525451 2.4307243 4.4128154 1.5241577 1.9693165 2.7768852 2.7075649
##  [43] 4.1405158 2.9656929 1.3726110 1.9231995 2.1492984 2.7768852 2.3161592
##  [50] 1.3726110 4.0218900 4.2436836 3.0500261 1.7394339 3.5407708 3.7942123
##  [57] 2.3161592 2.6735550 3.5925447 3.1869458 4.1405158 2.6397275 2.1052758
##  [64] 2.6735550 1.6073041 3.9349418 4.3540950 1.6073041 2.2347067 3.4370666
##  [71] 2.8867731 3.5925447 4.2005023 2.4307243 2.8491399 3.8893109 2.2347067
##  [78] 3.6439747 2.4307243 4.1405158 4.2385239 3.4888704 2.2347067 1.9231995
##  [85] 3.7942123 2.0605133 3.6948541 2.7768852 2.3161592 2.7075649 3.2351429
##  [92] 3.3855824 3.0500261 3.0943053 2.7075649 1.8769693 3.6439747 4.4128154
##  [99] 2.4307243 3.5407708 3.8423633 2.6058953 2.7768852 2.8491399 1.5651625
## [106] 1.5241577 2.4670564 2.7419463 2.6058953 3.0943053 4.1405158 3.9349418
## [113] 2.3161592 4.1405158 3.1869458 1.8769693 3.1869458 2.1052758 1.3726110
## [120] 2.4670564 2.3935108 3.4370666 2.1492984 4.3030596 2.8491399 2.3161592
## [127] 4.3761904 2.6735550 2.8491399 3.1869458 2.3553393 1.8769693 3.7942123
## [134] 2.5026075 1.6504822 2.7075649 3.4370666 2.6735550 3.0943053 3.4370666
## [141] 3.9349418 3.9349418 3.3855824 3.2844260 3.2351429 3.6948541 2.9656929
## [148] 1.5651625 2.7768852 2.2347067 2.7075649 1.7849035 3.0500261 2.6058953
## [155] 3.7942123 2.5026075 2.5375003 2.6735550 4.2436836 1.9231995 3.3855824
## [162] 2.8491399 3.6948541 3.8893109 5.1545360 2.7075649 2.0151431 1.6073041
## [169] 3.8423633 2.9656929 2.9656929 2.8125607 2.2347067
```

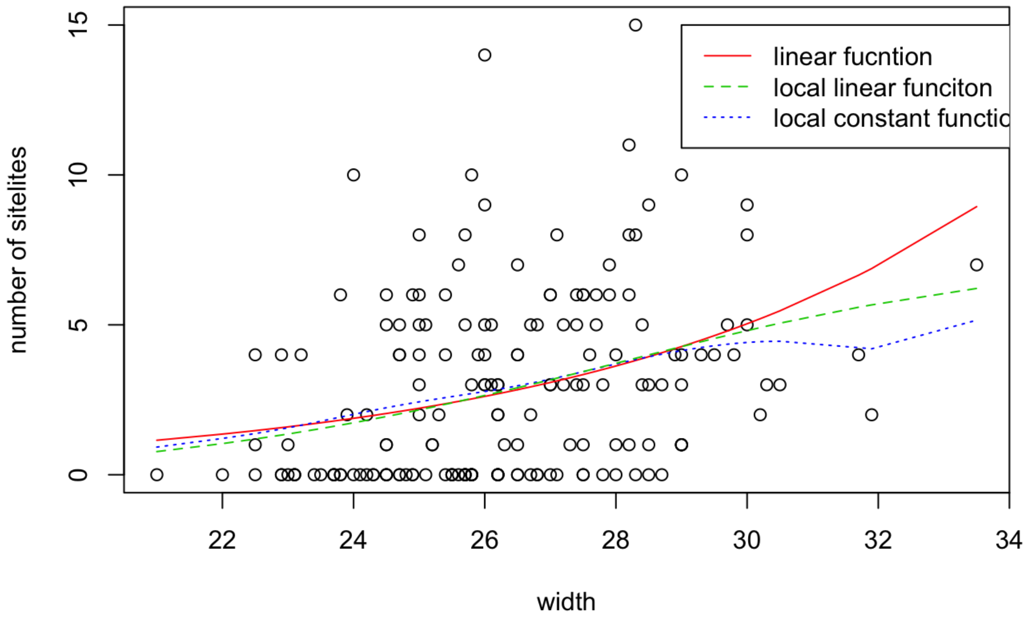
```
pos_bindwith_const
```

```
## [1] 1.206288
```

```
pos_const_data = cbind(crab$x,POS_CONST)
pos_const_data = pos_const_data[order(pos_const_data[,1]),]
```

partc

```
fit = glm(crab[,2]~crab[,1],family =poisson(link = "log"))
plot(crab[,1],crab[,2],xlab ="width",ylab ="number of sitelites")
prob2 = function(t){
  exp(fit$coef[1]+fit$coef[2]*t)
}
normal_possion_regression = cbind(crab[,1],prob2(crab[,1]))
normal_possion_regression = normal_possion_regression[order(normal_possion_regression[,1]),]
lines(normal_possion_regression[,1],normal_possion_regression[,2],col ="2",lty =1)
lines(pos_const_data[,1],pos_const_data[,2],col ="4",lty =3 )
lines(pos_linear_data[,1],pos_linear_data[,2],col ="3",lty = 2)
legend(29,15,c("linear fucntion","local linear funciton","local constant function"),col=c(2,3,4),lty=c(1,2,3
))
```



the assumption is reasonable

because overall, those three lines show same pattern except some distortion on the right side.