Kernel Smoother and Cross Validation

B.

```
compute_kernel <- function(d,h){
  ##Input: d- distance (scalar), h-bandwith (scalar)
  ##Output: kernel value (scalar)
  return(1/h*dnorm(d/h,mean=0,sd=1))
}

get_weight <- function(x_vec,new_x,ker_func,h){
  ##Input: x_vec- observation (vector), new_x - new x value (scalar), ker_func - function to compute ker
  ##Output: return weights (vector)
  dist_x <- x_vec-new_x
  return(ker_func(dist_x,h))
}
```

```
##Generate observations x and y
##Add white noise to the true function
n <- 100
x <- runif(n,-2,2)
y <- x^3 + rnorm(n,0,1)
```

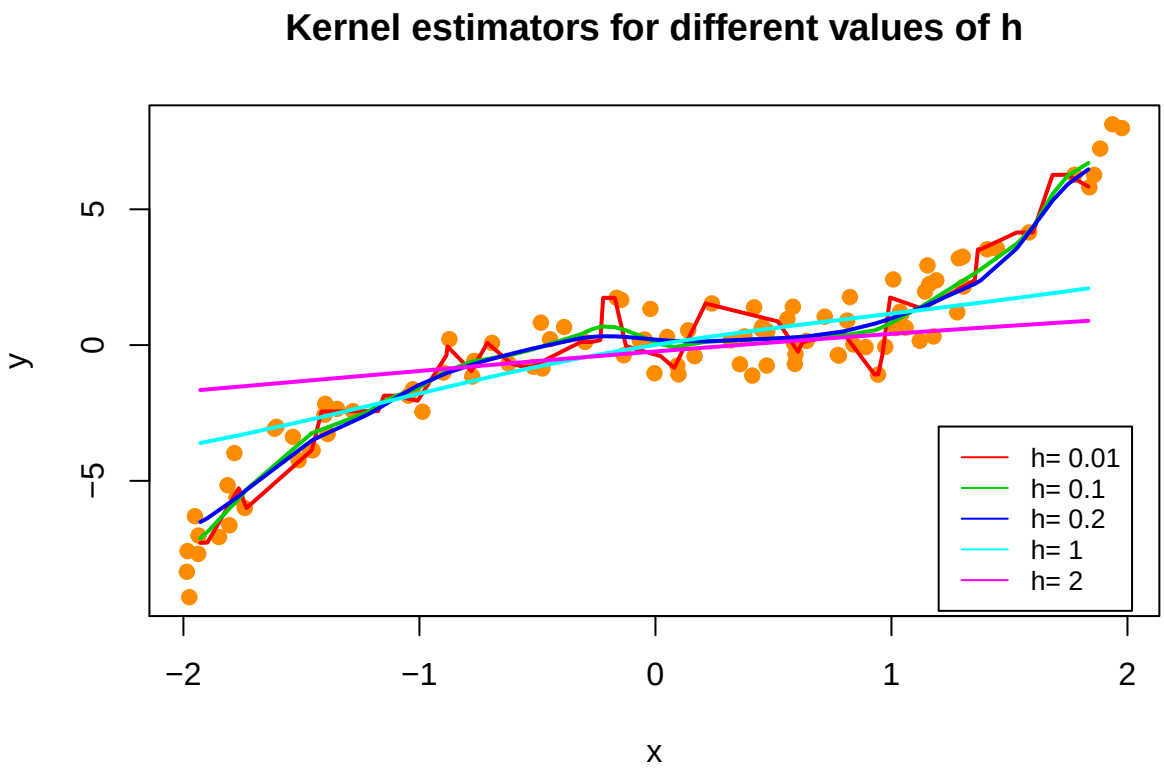
```
h_vec <- c(0.01,0.1,0.2,1,2)
B <- n/2
x_star <- runif(B,-2,2)
```

```
get_y_pred<- function(x,x_star,y,h){
  ##Input: x- observations (vector), x_star - new observations (vector), y-observations (vector), h - ba
  ##Output: y_pred - predictions of y valued (vector)
  W <- matrix(NA,nrow=length(x),ncol=length(x_star))
  for(i in 1:length(x_star)){
    W[,i]<-get_weight(x,x_star[i],compute_kernel,h)
  }

  normalize <- function(x){
    return(x/sum(x))
  }
  W_n <- apply(W,2,normalize)
  y_pred <- rep(NA,length(x_star))
  for(i in 1:length(x_star)){
    y_pred[i] <-sum(W_n[,i]*y)
  }
  return(y_pred)
}
```

```
##Create a matrix to store predictions for each value of h
Y_P <- matrix(NA,nrow=B,ncol=length(h_vec))
for(j in 1:length(h_vec)){
  Y_P[,j]<-get_y_pred(x,x_star,y,h_vec[j])
}
```

```
plot(x,y,main="Kernel estimators for different values of h", col="darkorange", pch=19)
col_vec <- 1:length(h_vec)+1
```

```
for(k in 1:length(h_vec)){
  x_idx <- order(x_star)
  lines(x_star[x_idx],Y_P[,k][x_idx],col=col_vec[k],lwd=2)
}
legend(1.2,-3, legend=paste("h=",h_vec),
       col=col_vec, lty=1, cex=0.8)
```



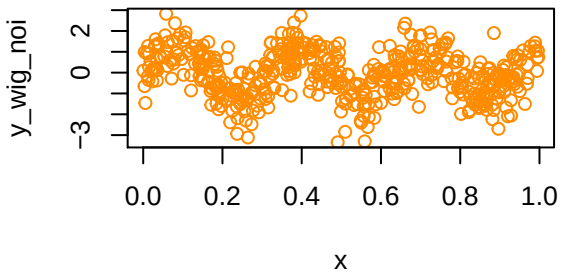
Cross validation

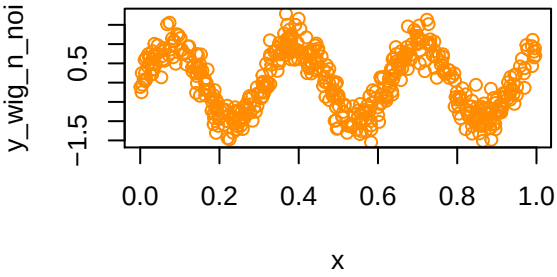
```
df <- cbind(x,y)

train_idx <- sample(1:n,0.8*n)
train <-  df[train_idx,]
test <- df[-train_idx,]
h <- 0.01
```

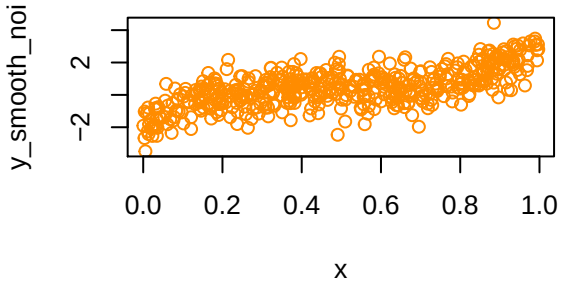
```
get_y_pred<- function(x_train,x_star,y_train,h){
  ##Input: take vectors x and y from training data and (x_star) from testing data
  ##Return: Predicted values of new y
  W <- matrix(NA,nrow=length(x_train),ncol=length(x_star))
  for(i in 1:length(x_star)){
    W[,i]<-get_weight(x_train,x_star[i],compute_kernel,h)
  }

  normalize <- function(x){
```

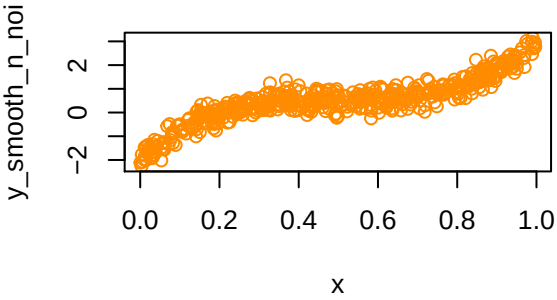
```
    return(x/sum(x))
  }
  W_n <- apply(W,2,normalize)
  y_pred <- rep(NA,length(x_star))
  for(i in 1:length(x_star)){
    y_pred[i] <-sum(W_n[,i]*y_train)
  }
  return(y_pred)
}
get_sse <- function(y_pred,y_star){
  ##Input - y_pred- predicted values of y (vector), y_star - new values of y (vector)
  ##Output: sum of square error
  return(sum((y_star-y_pred)^2))
}
get_err <- function(y_star,y_pred){
  return(y_star-y_pred)
}

get_pred_err <- function(x_train,x_star,y_train,y_star,h){
  ##Input: take vectors x and y from training data and (x_star) from testing data
  ##Return: Predicted values of new y and errors in a matrix where the first column is the   error and
  y_pred <-get_y_pred(x_train,x_star,y_train,h)
  sse <- get_sse(y_pred,y_star)
  err_pred <- cbind(y_star-y_pred,y_pred)
  return(err_pred)
}
```

```
n <- 500
train_idx <- sample(1:n,size=0.7*n)
test_idx <- seq(1:n)[-train_idx]
x <- runif(n,0,1)
not_noisy <- rnorm(n,0,0.3)
noisy <- rnorm(n,0,0.8)
y_wig_noi <- sin(20*x)+noisy
y_wig_n_noi <- sin(20*x)+not_noisy
y_smooth_noi <- 20*(x-0.5)^3+0.5 + noisy
y_smooth_n_noi <- 20*(x-0.5)^3+0.5 + not_noisy
par(mfrow=c(2,2))
plot(x,y_wig_noi,col="darkorange")
plot(x,y_wig_n_noi,col="darkorange")
plot(x,y_smooth_noi,col="darkorange")
plot(x,y_smooth_n_noi,col="darkorange")
```










```
h_vec <- seq(0.01,0.1,0.001)
x_train <- x[train_idx]
x_test <- x[test_idx]
y_pred_w_noi <- get_y_pred(x_train,x_test,y_wig_noi[train_idx],0.01)
y_pred_w_n_noi <- get_y_pred(x_train,x_test,y_wig_n_noi[train_idx],0.01)
y_pred_s_noi <- get_y_pred(x_train,x_test,y_smooth_noi[train_idx],0.01)
y_pred_s_n_noi <- get_y_pred(x_train,x_test,y_smooth_n_noi[train_idx],0.01)

get_test_err <- function(x_train,x_test,y_train,y_test, h_vec){
  err_vec <- rep(NA,length(h_vec))
  for(i in 1:length(h_vec)){
  y_pred <- get_y_pred(x_train,x_test,y_train,h_vec[i])
  err_vec[i] <- get_sse(y_pred,y_test)
  }
  return(err_vec)
}
t_err_w_noi <-get_test_err(x_train,x_test,y_wig_noi[train_idx],y_wig_noi[test_idx],h_vec)
t_err_w_n_noi <-get_test_err(x_train,x_test,y_wig_n_noi[train_idx],y_wig_n_noi[test_idx],h_vec)
t_err_s_noi <-get_test_err(x_train,x_test,y_smooth_noi[train_idx],y_smooth_noi[test_idx],h_vec)
t_err_s_n_noi<-get_test_err(x_train,x_test,y_smooth_n_noi[train_idx],y_smooth_n_noi[test_idx],h_vec)

best_h_w_noi <- which.min(t_err_w_noi)
best_h_w_n_noi <- which.min(t_err_w_n_noi)
best_h_s_noi <- which.min(t_err_s_noi)
best_h_s_n_noi <- which.min(t_err_s_n_noi)
```

```
y_pred_w_noi <- get_y_pred(x_train,x_test,y_wig_noi[train_idx],h_vec[best_h_w_noi])
y_pred_w_n_noi <- get_y_pred(x_train,x_test,y_wig_n_noi[train_idx],h_vec[best_h_w_n_noi])
y_pred_s_noi <- get_y_pred(x_train,x_test,y_smooth_noi[train_idx],h_vec[best_h_s_noi])
y_pred_s_n_noi <- get_y_pred(x_train,x_test,y_smooth_n_noi[train_idx],h_vec[best_h_s_n_noi])
```

```
x_idx <- order(x_test)
par(mfrow=c(2,2))
plot(x,y_wig_noi,col="darkorange",pch=19)
lines(x_test[x_idx],y_pred_w_noi[x_idx],col="cyan4",lwd=3)
legend(0.6,-1.5, legend=paste("h=",h_vec[best_h_w_noi]),
       col="cyan4", lty=1, cex=0.8)
plot(x,y_wig_n_noi,col="darkorange",pch=19)
lines(x_test[x_idx],y_pred_w_n_noi[x_idx],col="cyan4",lwd=3)
legend(0.6,-1, legend=paste("h=",h_vec[best_h_w_n_noi]),
       col="cyan4", lty=1, cex=0.8)
plot(x,y_smooth_noi,col="darkorange",pch=19)
lines(x_test[x_idx],y_pred_s_noi[x_idx],col="cyan4",lwd=3)
legend(0.6,-1, legend=paste("h=",h_vec[best_h_s_noi]),
       col="cyan4", lty=1, cex=0.8)
plot(x,y_smooth_n_noi,col="darkorange",pch=19)
lines(x_test[x_idx],y_pred_s_n_noi[x_idx],col="cyan4",lwd=3)
legend(0.6,-1, legend=paste("h=",h_vec[best_h_s_n_noi]),
       col="cyan4", lty=1, cex=0.8)
```