HW4

Thibaud Bruyelle

10/30/2020

Question 7

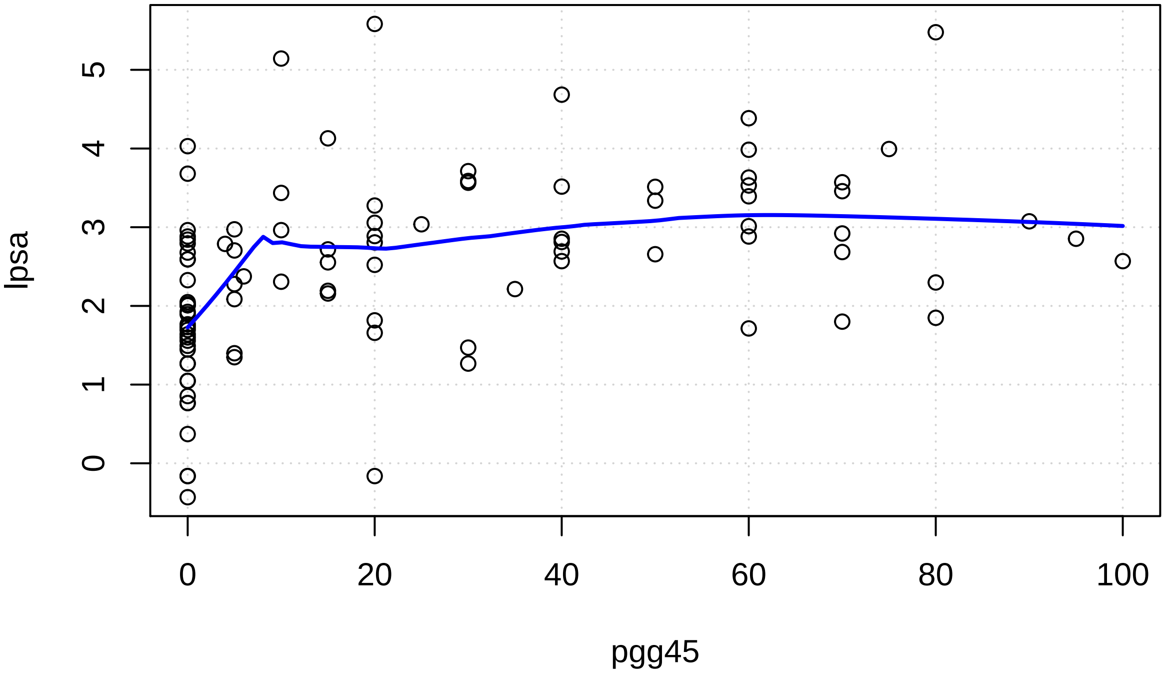
```r
# weight function
W <- function(r){
  res <- ifelse(abs(r)<1, (1-abs(r)^3)^3, 0)
  return(res)
}
```

```r
# the following function computes the local linear regression fit in x0
llr <- function(x, y, x0, span, hat = FALSE){

  n = length(x)
  # compute w_s
  k = floor(span * n)
  w_s = sort(abs(x-x0))[k] # never compute xk
  # FIXME : what if w_s = 0?

  # compute the model by computing each component of vec h(x0)
  m0 = sum(W(abs(x-x0)/w_s))
  n0 = sum(W(abs(x-x0)/w_s) * x)
  p0 = sum(W(abs(x-x0)/w_s) * x^2)
  delta0 = m0 * p0 - n0^2
  h = W(abs(x-x0)/w_s) * (1/delta0) * (p0 - x0*n0 + x*(m0 * x0 - n0))


  if (hat){
    res = list("h" = h, "y0_fit" = t(h) %*% y)
  }
  else{
    res = list("y0_fit" = t(h) %*% y)
  }
  return(res)
}
```

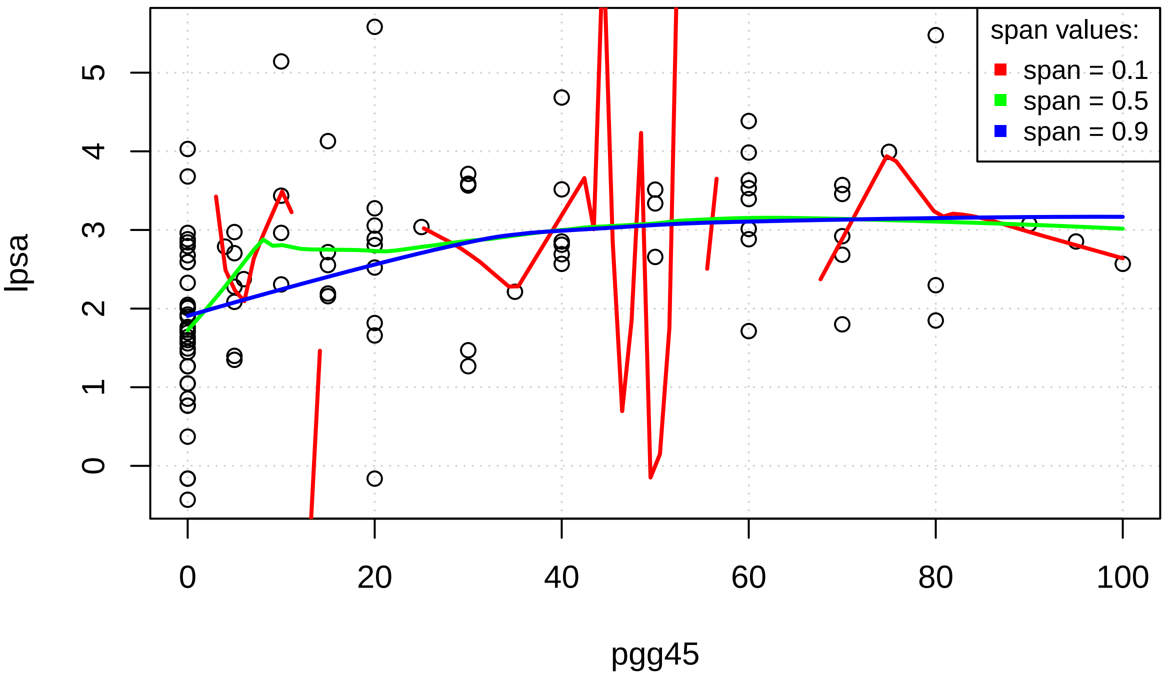
Question 8

```r
# import data
p <- '/Users/thibaudbruyelle/Documents/Stanford/fall2020/stats305A/datasets/lprostate.dat'
data <- read.table(p, header = T)[,-1]
x = data$pgg45
y = data$lpsa
n = length(x)
```

```
# run llr
span = 0.5
x0_seq = seq(min(x), max(x), length.out = 100)
y_fit = c()
for (i in 1:length(x0_seq)){
  mod_i = llr(x, y, x0_seq[i], span)
  y_fit = c(y_fit, as.numeric(mod_i[[1]]))
}

plot(x,y, xlab = 'pgg45', ylab = 'lpsa', panel.first=grid())
lines(x0_seq, y_fit, col = 'blue', lwd = 2)
```



```
generate_y_fit <- function(x, y, x0_seq, span){
  y_fit = c()
  for (i in 1:length(x0_seq)){
    mod_i = llr(x, y, x0_seq[i], span)
    y_fit = c(y_fit, as.numeric(mod_i[[1]]))
  }

  return(y_fit)
}

y_fit_span_1 = generate_y_fit(x, y, x0_seq, 0.1)
y_fit_span_2 = generate_y_fit(x, y, x0_seq, 0.5)
y_fit_span_3 = generate_y_fit(x, y, x0_seq, 0.9)

plot(x,y, xlab = 'pgg45', ylab = 'lpsa', panel.first=grid())
lines(x0_seq, y_fit_span_1, col = 'red', lwd = 2, type ='l')
lines(x0_seq, y_fit_span_2, col = 'green', lwd = 2)
lines(x0_seq, y_fit_span_3, col = 'blue', lwd = 2)
legend('topright', legend = c("span = 0.1", "span = 0.5", "span = 0.9"),
       col = c("red", "green", "blue"), title = 'span values: ', cex = 0.8, pch = 15)
```

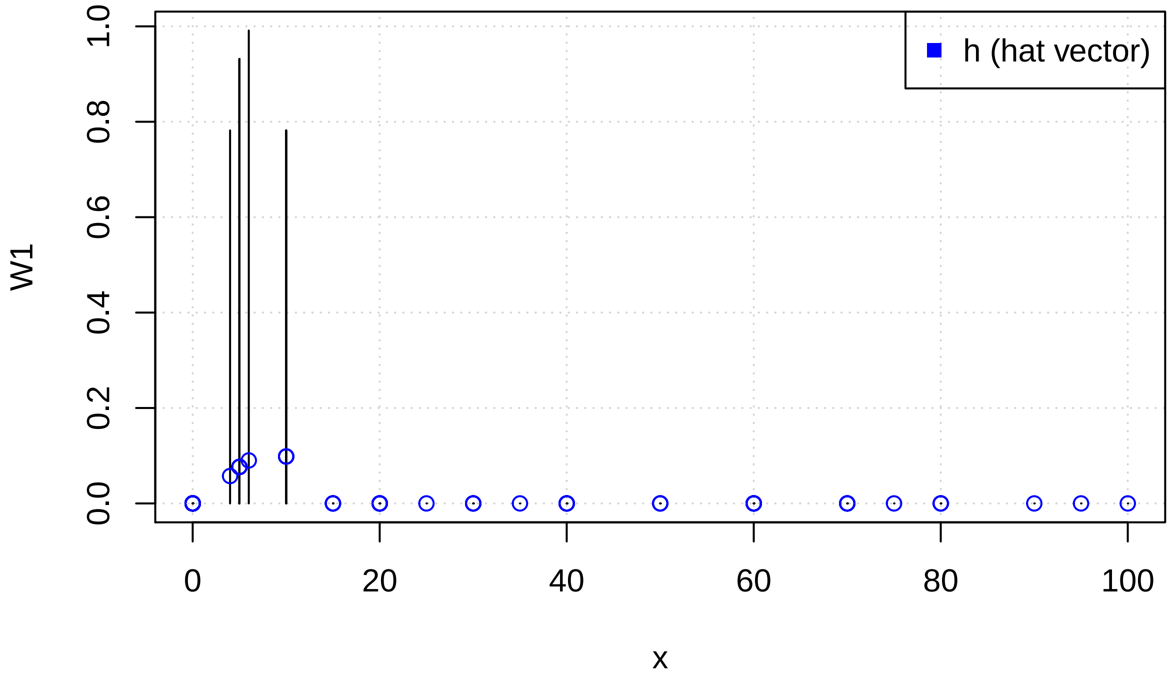


```
# lines(x0_seq, y_fit_span_1, col = 'blue', lwd = 2)
```

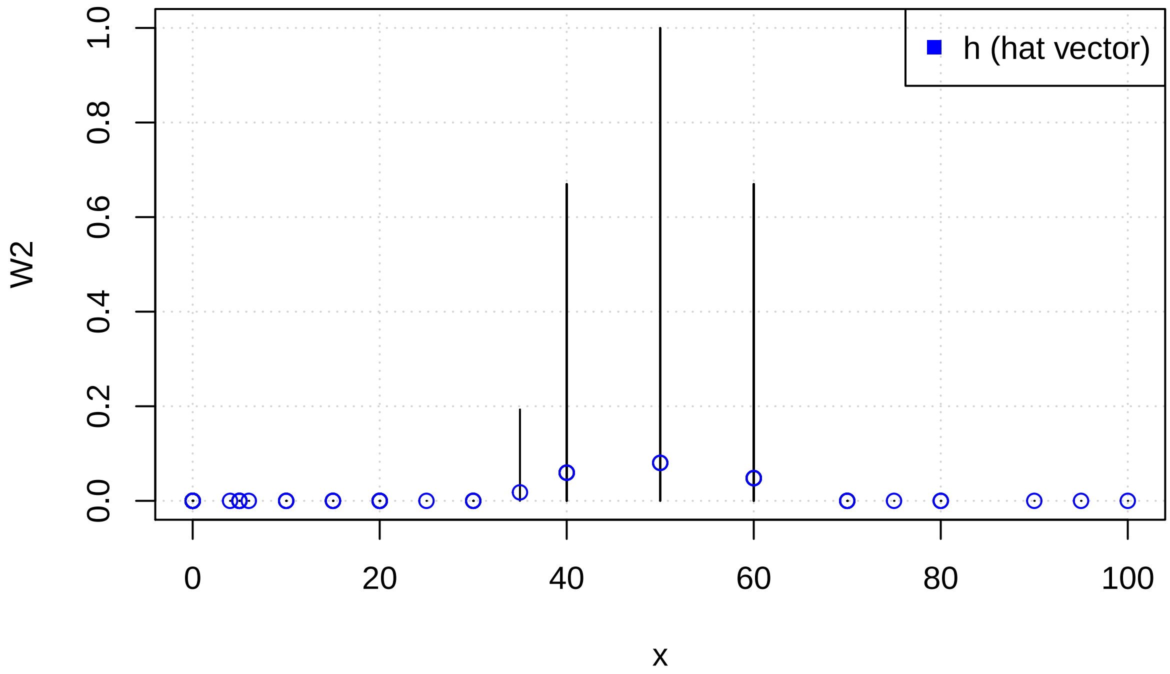
After having tested multiple values of `span` within the range $[0, 1]$, `span = 0.5` seems to be a good value. We will provide more details about the span selection in question 10.

Question 9

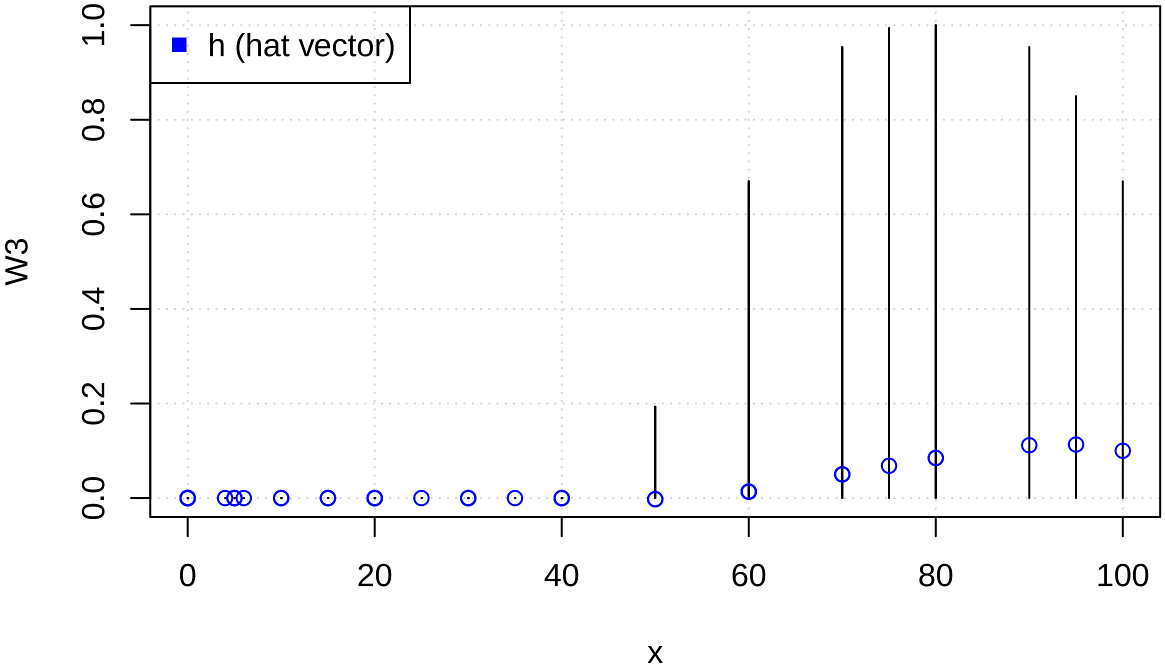
```
span = 0.25
k = floor(span * n)
# Select 3 values of x0
x01 = 7
w_s1 = sort(abs(x-x01))[k]
x02 = 50
w_s2 = sort(abs(x-x02))[k]
x03 = 80
w_s3 = sort(abs(x-x03))[k]

# Compute weights
W1 = W(abs(x-x01)/w_s1)
W2 = W(abs(x-x02)/w_s2)
W3 = W(abs(x-x03)/w_s3)
# Compute weight vector h
h1 = llr(x, y, x01, span, TRUE)$h
h2 = llr(x, y, x02, span, TRUE)$h
h3 =  llr(x, y, x03, span, TRUE)$h

# par(mfrow = c(3,1))
plot(x, W1, panel.first=grid(), type = 'h')
lines(x, h1, type = 'p', col = 'blue')
legend('topright', legend = c('h (hat vector)'), col = c('blue'), pch =15)
```



```
plot(x, W2, panel.first=grid(), type = 'h')
lines(x, h2, type = 'p', col = 'blue')
legend('topright', legend = c('h (hat vector)'), col = c('blue'), pch =15)
```



```
plot(x, W3, panel.first=grid(), type = 'h')
lines(x, h3, type = 'p', col = 'blue')
legend('topleft', legend = c('h (hat vector)'), col = c('blue'), pch =15)
```



Question 10

Let us assume that we have a set of span values S (for instance we could have $S =]0, 1]$ in our case) and we want to find a systematic way for selecting the most appropriate span value. On the one hand, we do not want the span to be too small because it would imply $w_s = 0$ and it would yield in an undefined model at x_0. On the other hand we also want to avoid big span values because it would mean that w_s will be big and thus all weights will be nonzeros: somehow we would lost the *local* properties of the model. In order to pick the best possible value for the span, one could try some validation techniques. For instance, for each value s, $s \in S$, we could compute the model `llr(x, y, x0, s)` for all `x0` within `x` range. Then, we could assess the performace of the whole model by computing the prediction error (as we know `y`): `error_s`. It would be a measure of the average performance at all `x` with respect to s. Then we could select the value of s that minimizes this error.

Also, a pratical way for selecting the span is to look at the hat matrix $\boldsymbol{H}$. If the span value is small enough, it is possible to have a lot of `NA` entries in the matrix. As we want to avoid this situation, we should choose a span value that is big enough.

Question 11

Our model prediction vector writes:

$$\widehat{\boldsymbol{y}} = \boldsymbol{H}\boldsymbol{y}$$

where the rows of $\boldsymbol{H}$ are the $\boldsymbol{h}^T(x_i)$ with x_i is the i$-th$ component of $\boldsymbol{x}$. One way to measure the effective degrees of freedom is to compute $tr(\boldsymbol{H})$. Indeed, this metric accumulates fractional degrees of freedom for directions of $\boldsymbol{y}$ that are shrunk, but not entirely eliminated, in computing $\widehat{\boldsymbol{y}}$.

Question 12

`llr_wrapper` will compute the estimated error variance that will be estimated by:

$$\boxed{\hat{\sigma}^2 = \frac{1}{n}\sum_{i=1}^{n}(\widehat{\boldsymbol{y}}_i - \boldsymbol{y}_i)^2}$$

where $\widehat{\boldsymbol{y}} = \boldsymbol{H}\boldsymbol{y}$.

Then, we can estimate the standard error of the fit for each fitted value $\hat{f}(x_0)$. The true variance of the fit satisfies:

$$var(\hat{f}(x_0)) = var(\boldsymbol{h}(x_0)^T) = var(\sum_{i=1}^{n} y_i h_i(x_0))$$

As the $(y_i)_{1 \leq i \leq n}$ are *iid*, it yields to:

$$var(\hat{f}(x_0)) = \sum_{i=1}^{n} h_i(x_0)^2 \cdot \sigma^2$$

.

So finally:

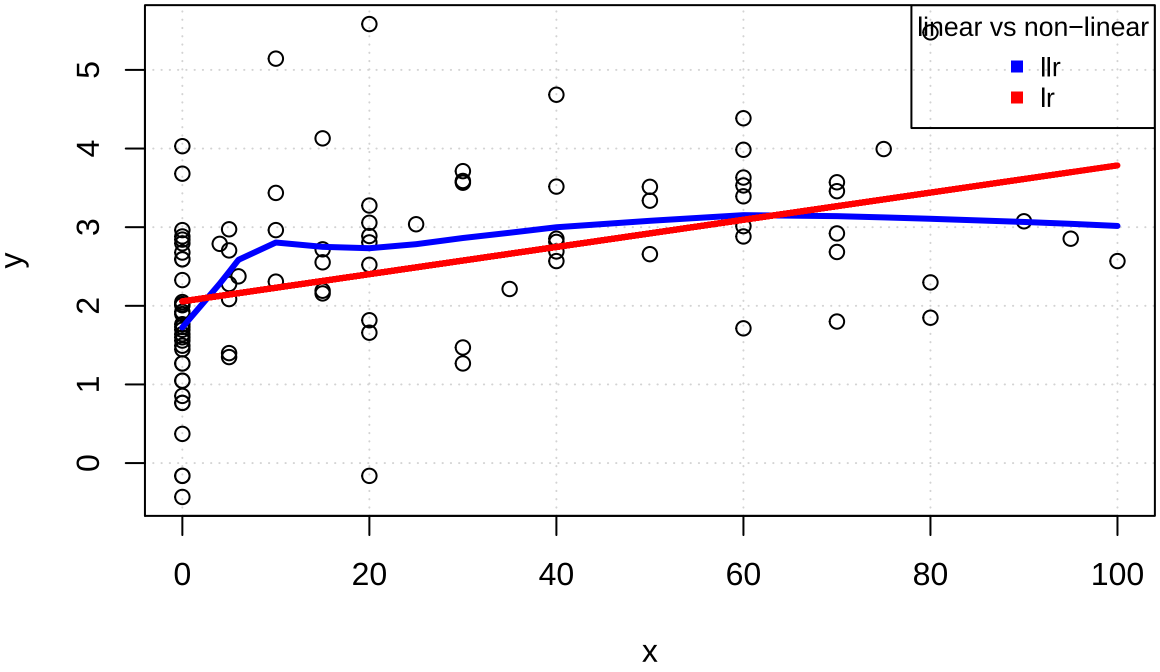
$$\boxed{\widehat{var(\hat{f}(x_0))} = \sum_{i=1}^{n} h_i(x_0)^2 \cdot \hat{\sigma}^2}$$

which the square value of the estimated standard error of the fit.

```r
llr_wrapper <- function(x, y, span, hat = FALSE){

  # x = unique(x)
  n = length(x)
  H = matrix(ncol = n, nrow = n) # the hat matrix that will be returned
  y_fit = c()

  for (i in 1:n){
    x0 = x[i]
    model_i <- llr(x, y, x0, span, TRUE) # we return the hat vector
    fitted_y <- as.numeric(model_i$y0_fit)
    y_fit =  c(y_fit, fitted_y) # update prediction vector
    h_x0 <- model_i$h
    H[i, ] = h_x0
  }

  error_estimated_variance = mean((y - y_fit)^2, na.rm = T)
  tr_H = sum(diag(H), na.rm = T)
  # Estimated standard error of the fit for each fitted value.
  se_vec = (error_estimated_variance * rowSums(H^2))^(0.5)

  # Return options
  if (hat){
    res = list("y_fit" = y_fit, "df" = tr_H, "hat" = H,
               "est_error_var" = error_estimated_variance, "se_vec" = se_vec)
  }
  else{
    res = list("y_fit" = y_fit, "df" = tr_H,
               "est_error_var" = error_estimated_variance, "se_vec" = se_vec)
  }

  return(res)
}
```

Question 13

```
# initial plot w/ span = 0.5
span = 1/2
model_llr <- llr_wrapper(x, y, span, TRUE)
# run linear regression
model_lr <- lm(y ~ x)
# superimpose the plots
plot(x, y, panel.first=grid())
# llr plot
tab <- data.frame('x' = unique(x), 'y' = unique(model_llr$y_fit))
tab <- arrange(tab, x)
lines(tab$x, tab$y, type = 'l',col = 'blue', lwd = 3)
# linear regression plot
lines(x, model_lr$coefficients[1] + model_lr$coefficients[2] * x, col = 'red', lwd = 3)
legend('topright', legend = c('llr', 'lr'), col = c('blue', 'red'),
       pch = 15, title = 'linear vs non-linear', cex = 0.8)
```



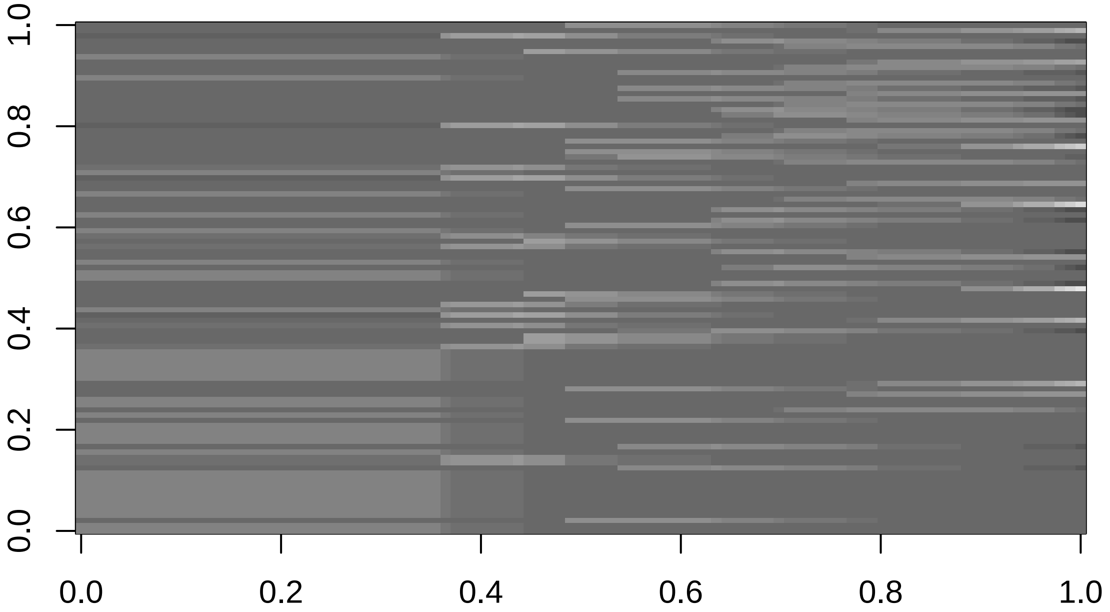
An informal F-test could be:

$$F := \frac{(RSS_{non\_linear} - RSS_{linear})/df(llr)}{RRS_{linear}/df(lr)} \sim F(df(llr), n-2)$$

## Question 14

```
H_hat <- model_llr$hat
# reorder the rows of H_hat with respect to `pgg45` increasing order
perm_dat <- data.frame("row_id" = c(1:length(x)), "x" = x)
perm_dat = arrange(perm_dat, x)
# perm_dat <- perm_dat[duplicated(perm_dat),]
rows_perm <- perm_dat$row_id
# rows_perm <- c(1, 57, 14, 42, 38, 59, 72, 83, 39, 48, 51, 71, 53, 90, 29, 74, 63, 47)
H_hat <- H_hat[rows_perm, ]
# ouptut image matrix
```

```r
# H_hat <- H_hat[unique(H_hat),]
# FIXME: how do we interpret it?
image(H_hat, col = gray.colors(33))
```



Question 15