UK Complex Statistical Methods, WS 2024/25
Exercise Sheet 3

Khodosevich Leonid

16.01.2025

Problems

Problem 1

1. B-splines defined on a uniform knot sequence $(\dots, \tau_{-1}, \tau_0, \tau_1, \dots) = (\dots, -1, 0, 1, \dots) = \mathbb{Z}$ are called cardinal B-splines and denoted by N_i^c, m. For a given m prove that

$$N_i^c, m(x) = \frac{(-1)^m}{(m-1)!} \sum_{j=0}^{m} (-1)^{m-j} \binom{m}{j} (x - i - j)_+^{m-1}.$$

Hint: first prove by induction the equality

$$n![0, 1, \dots, n]f = \sum_{i=0}^{n} (-1)^{n-i} \binom{n}{i} f(i),$$

and then employ the definition

$$N_i(x) = (-1)^m (\tau_{i+m} - \tau_i)[\tau_i, \dots, \tau_{i+m}](x - \cdot)_+^{m-1}.$$

(3 points)

SOLUTION

1) Base of induction: $0![0]f = (-1)\binom{0}{0} f(0)$, which yields $f(0) = f(0)$, which is true. 2) Assume it holds for m. Lets prove for m+1

$$\begin{aligned}(n+1)![0,...,n+1]f &= \frac{(n+1)![1,...,n+1]f-(n+1)![0,...,n]f}{n+1}\\ &= n![1,...,n+1]f-n![0,...,n]f \text{ (using def of divided difference)}\\ &= \sum_{i=1}^{n+1}(-1)^{n-(i-1)}\binom{n}{i-1}f(i)-\sum_{i=0}^{n}(-1)^{n-i)}\binom{n}{i}f(i)\\ &= \sum_{i=0}^{n}(-1)^{n-i}\binom{n}{i}f(i+1)-\sum_{i=0}^{n}(-1)^{n-i)}\binom{n}{i}f(i)\\ &= \sum_{i=0}^{n}(-1)^{n-i)}\binom{n}{i}(f(i+1)-f(i)) = (*)\end{aligned}$$

Now, expand (*) to see how it works

$$\begin{aligned}(*) &= (-1)^n\binom{n}{0}(f(1)-f(0))-(-1)^{n-1}\binom{n}{1}(f(2)-f(1))+...+(f(n+1)-f(n))\\ &= (-1)^{n+1}\binom{n}{0}f(0)+(-1)^n(\binom{n}{1}+\binom{n}{0})f(1)+...+f(n+1)\\ &= |\text{Now we use that}\binom{n}{k}=\binom{n-1}{k-1}+\binom{n-1}{k}\text{and}\binom{n}{0}=\binom{n+1}{0},\binom{n}{n}=\binom{n+1}{n+1}|\\ &= (-1)^{n+1}\binom{n+1}{0}f(0)+(-1)^n\binom{n+1}{1}f(1)+...+\binom{n+1}{n+1}f(n+1)\\ &= \sum_{i=0}^{n+1}(-1)^{n+1-i}\binom{n+1}{i}f(i)\end{aligned}$$

Which is exactly what we wanted to show.
2) Now, apply that to definition of B-spline:

$$\begin{aligned}N_i(x) &= (-1)^m(\tau_{i+m}-\tau_i)[\tau_i,\ldots,\tau_{i+m}](x-\cdot)_+^{m-1}.\\ &= \frac{(-1)^m}{(m-1)!}\sum_{j=0}^{m}(-1)^{m-j}\binom{m}{j}(x-i-j)_+^{m-1}.\end{aligned}$$

Which is exactly what we wanted to show.

■

2. Show that

$$f(x)=\sum_{i=0}^{m-1}\frac{x^i}{i!}f^{(i)}(0)+\int_0^1\frac{(x-u)_+^{m-1}}{(m-1)!}f^{(m)}(u)\,du.$$

Hint: start by writing $f(x)=\int_0^x f'(u)\,du+f(0)$. (2 points)

SOLUTION

Taylor expression for a function with integral remainder:
$f(x) = f(a) + f'(a)(x-a) + \frac{f''(a)}{2!}(x-a)^2 + \cdots \frac{f^{(m-1)}(a)}{(m-1)!}(x-a)^{m-1} + \int_a^x f^{(m)}(u)\frac{(x-u)^m}{m!}du.$
Written for a = 0:

$$\begin{aligned}
f(x) &= f(a) + f'(a)(x-a) + \frac{f''(a)}{2!}(x-a)^2 + \cdots \frac{f^{(m-1)}(a)}{(m-1)!}(x-a)^{m-1} + \int_a^x f^{(m)}(u)\frac{(x-u)^{m-1}}{(m-1)!}du \\
&= f(0) + f'(0)(x) + \frac{f''(0)}{2!}(x)^2 + \cdots \frac{f^{(m-1)}(0)}{(m-1)!}(x)^{m-1} + \int_0^x f^{(m)}(u)\frac{(x-u)^{m-1}}{(m-1)!}du+ \\
&= \sum_{i=0}^{m-1} \frac{x^i}{i!} f^{(i)}(0) + \int_0^x f^{(m)}(u)\frac{(x-u)^{m-1}}{(m-1)!}du \\
&= \sum_{i=0}^{m-1} \frac{x^i}{i!} f^{(i)}(0) + \int_0^1 f^{(m)}(u)\frac{(x-u)_+^{m-1}}{(m-1)!}du
\end{aligned}$$

In last step we used the fact that we normalize x to [0, 1] and def of (x - u)$_+$ as if integral is within [0, 1] then $(x-u)_+ = (x-u)$

■

3. Let $\{\phi_i\}_{i=1}^{\infty}$ be the trigonometric Fourier basis in $L^2[0,1]$, that is

$$\phi_1(x) = 1, \quad \phi_{2i}(x) = \sqrt{2}\cos(2\pi i x), \quad \phi_{2i+1}(x) = \sqrt{2}\sin(2\pi i x), \quad i = 1, 2, \ldots$$

Define also a periodic Sobolev class

$$W_{2,\text{per}}^m = \{f \in W_2^m : f^{(j)}(0) = f^{(j)}(1),\ j = 0, 1, \ldots, m-1\}.$$

In particular, any $f \in W_{2,\text{per}}^m$ can be represented as

$$f(x) = \sum_{i=1}^{\infty} \theta_i \phi_i(x) = \theta_1 + \sum_{k=1}^{\infty} \{\theta_{2k}\phi_{2k}(x) + \theta_{2k+1}\phi_{2k+1}(x)\},$$

where θ_i are the corresponding Fourier coefficients. For $f \in W_{2,\text{per}}^2$, derive the expression of $\int_0^1 \{f''(x)\}^2\,dx$ in terms of Fourier coefficients of f. (2 points)

Excercise sheet 3 (problems 2, 3)

Khodosevich Leonid

Problem 2

Subtask 1.

Problem 2

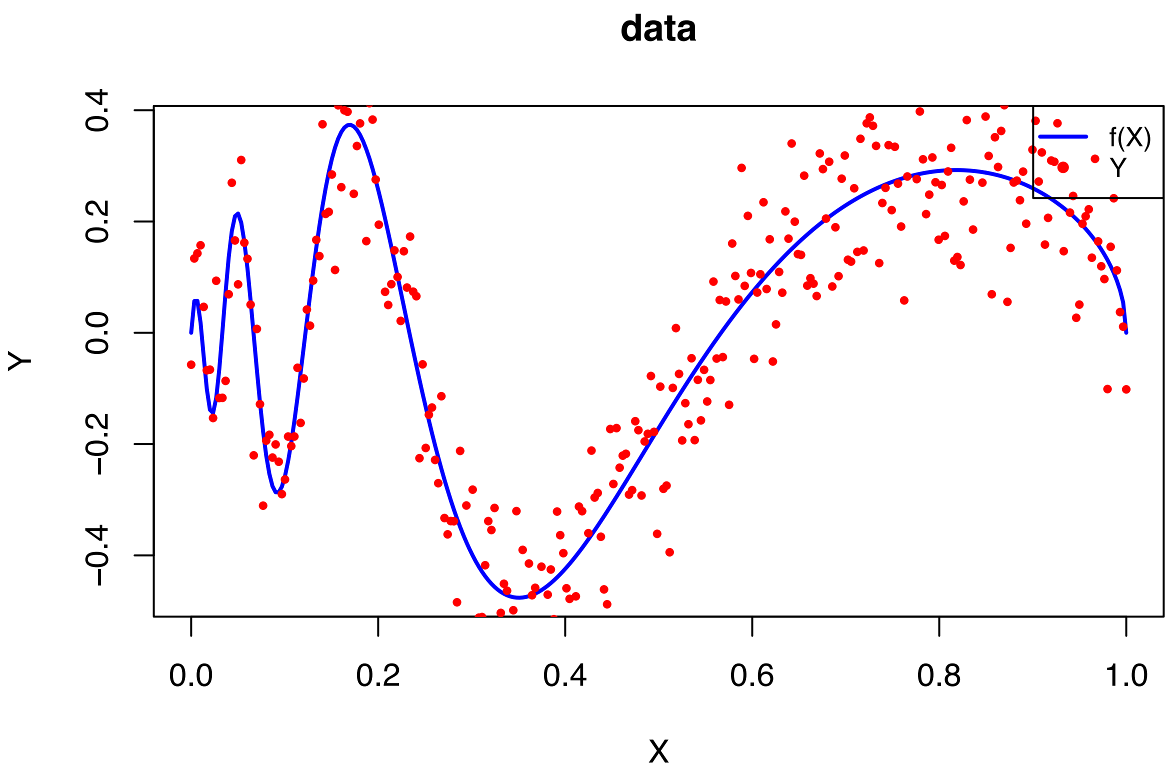
1. Let $X_1, \dots, X_{300}$ be equispaced points on $[0, 1]$. Set seed to 2425 and simulate Y_i according to the model

$$Y_i = f(X_i) + 0.1\varepsilon_i, \quad f(X_i) = \sqrt{x(1-x)}\sin\left(\frac{\pi}{1.5(x+0.1)}\right), \quad i = 1, \dots, 300,$$

with $\varepsilon_1, \dots, \varepsilon_{300} \overset{i.i.d.}{\sim} \mathcal{N}(0, 1)$.

```r
f <- function(x) {
  sqrt(x * (1 - x)) * sin(pi / (1.5 * (x + 0.1)))
}
X <- seq(0, 1, length.out = 300)
epsilon <- rnorm(300, mean = 0, sd = 1)
Y <- f(X) + 0.1 * epsilon
```

```r
plot(X, f(X), type = "l", col = "blue", lwd = 2,
     main = "data", xlab = "X", ylab = "Y")
points(X, Y, col = "red", pch = 16, cex = 0.6)

legend("topright", legend = c("f(X)", "Y"),
       col = c("blue", "red"), lty = c(1, NA), pch = c(NA, 16),
       lwd = c(2, NA), cex = 0.8)
```

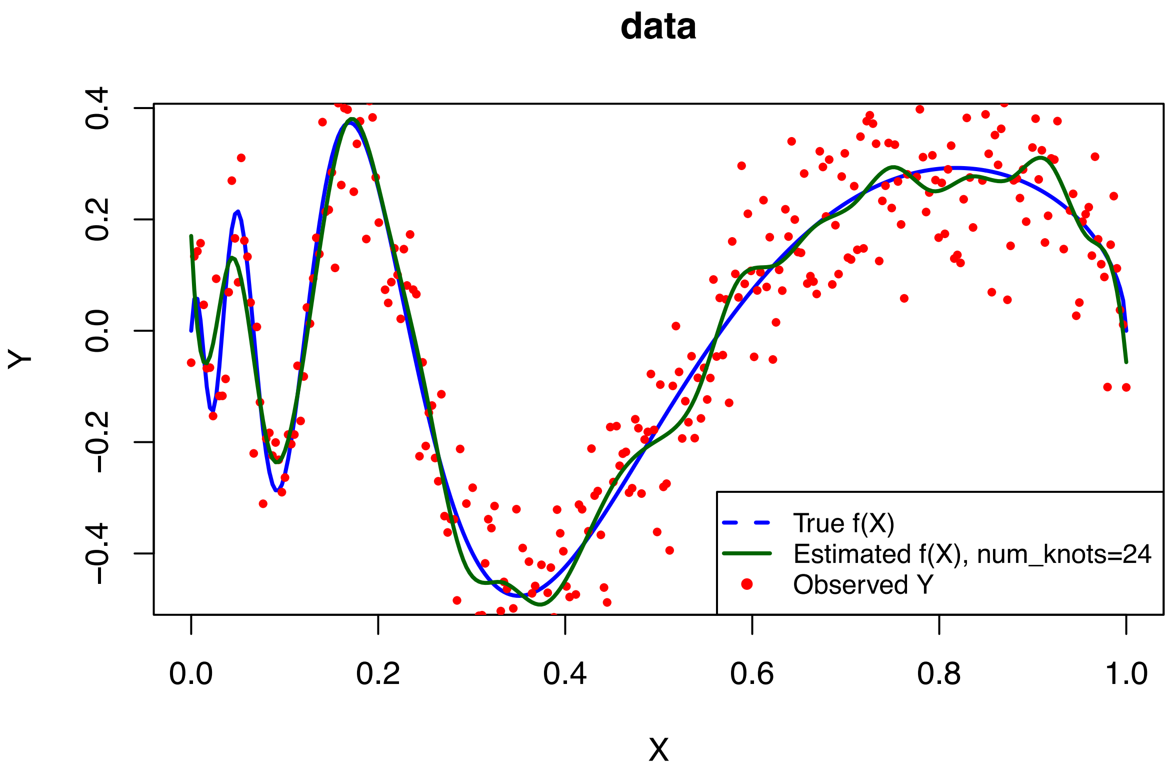


(a) Estimate f using a least-squares spline estimator of degree 3 and equidistant knots. Choose the number of equidistant knots by GCV (generalised cross-validation). Plot the obtained estimator and comment on the results. (3 points)

! Formula of GCV taken from https://epub.uni-regensburg.de/27968/1/DP472_Kagerer_introduction_splines.pdf

```
GCV <- function(X, Y, degree) {
  gcv <- function(ndx, X, Y, degree) {
    dx <- 1 / ndx
    knots <- seq(0 - degree * dx, 1 + degree * dx, by = dx)
    N <- spline.des(knots, X, degree + 1, rep(0, length(X)), outer.ok = TRUE)$design
    H <- solve(t(N) %*% N) %*% t(N)
    beta <- H %*% Y
    Y_hat <- N %*% beta
    residuals <- Y - Y_hat
    return(sum(residuals^2)/(1-sum(diag(H)/length(Y)))^2)
  }
  return(optimize(function(ndx) gcv(round(ndx),X=X,Y=Y,degree=degree),interval=c(2, 30))$minimum)
}

degree=3

opt_num_knots <- GCV(X, Y, degree)
opt_num_knots <- round(opt_num_knots)

ndx <- opt_num_knots
degree <- 3
dx <- 1/ndx
knots <- seq(0-degree*dx,1+degree*dx,by=dx)
N=spline.des(knots,X,degree+1,0*X,outer.ok=T)$design
```

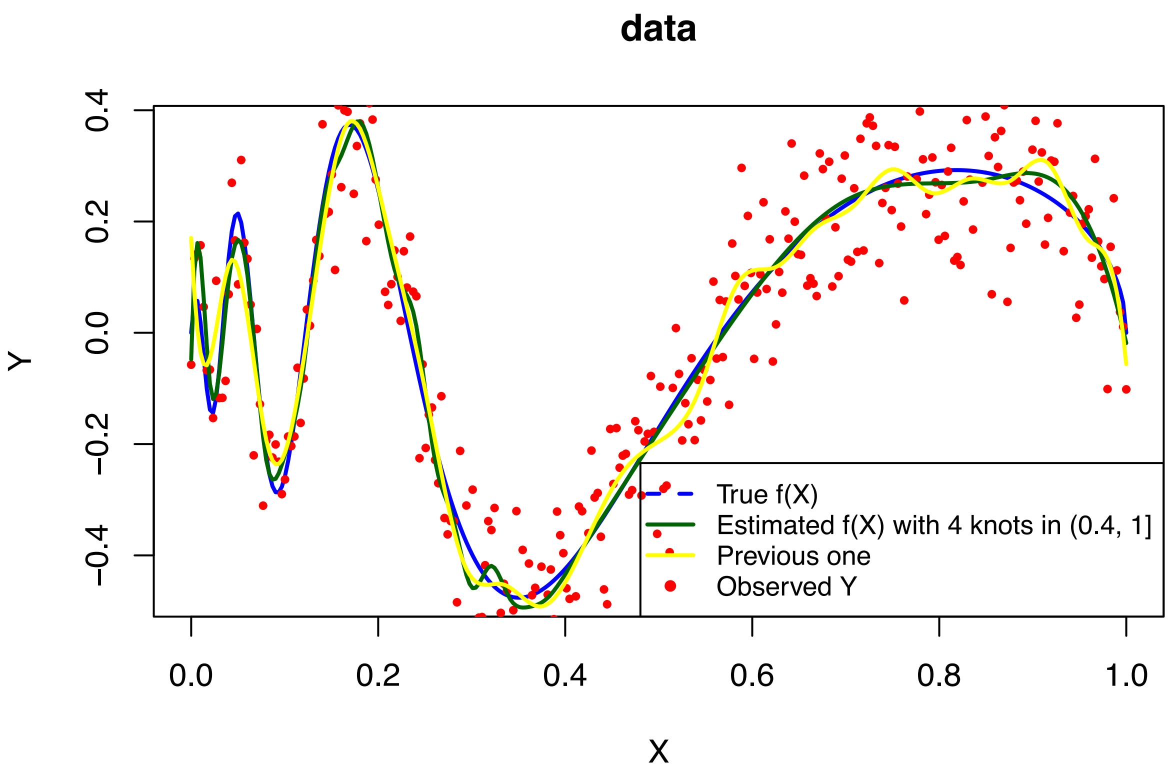
```r
H  <-  solve(t(N)%*%N)%*%t(N)
beta <- H%*%Y


plot(X, f(X), type = "l", col = "blue", lwd = 2,
     main = "data", xlab = "X", ylab = "Y")

points(X, Y, col = "red", pch = 16, cex = 0.6)

lines(X, N%*%beta, col = "darkgreen", lwd = 2)

legend("bottomright", legend = c("True f(X)", paste0("Estimated f(X), num_knots=",opt_num_knots), "Obse
       col = c("blue", "darkgreen", "red"), lty = c(2, 1, NA), pch = c(NA, NA, 16),
       lwd = c(2, 2, NA), cex = 0.8)
```



Model badly estimates data in interval (0, 0.2) as it's changing too frequently Idea would be using higher amount of knots, but that would increase variance in (0.2 1). 1b suggests use different amount of knots on different intervals.

(b) Modify the estimator from (a) such that you have only four knots on the interval $(0.4, 1]$. Add this new estimator to the plot from (a) and comment on the result. (2 points)

```r
GCV <- function(X, Y, degree) {
  gcv <- function(ndx, X, Y, degree) {
    dx <- 1 / ndx
    knots <- seq(0 - degree * dx, 1 + degree * dx, by = dx)
    N <- spline.des(knots, X, degree + 1, rep(0, length(X)), outer.ok = TRUE)$design
    H <- solve(t(N) %*% N) %*% t(N)
    beta <- H %*% Y
    Y_hat <- N %*% beta
    residuals <- Y - Y_hat
    return(sum(residuals^2)/(1-sum(diag(H)/length(Y)))^2)
  }
  return(optimize(function(ndx) gcv(round(ndx),X=X,Y=Y,degree=degree),interval=c(2, 30))$minimum)
}


degree <- 3

ndx<-4
dx<-(1 - 0.4)/ndx
int_knots<-seq(0.4-degree*dx,1+degree*dx,by=dx)
int_knots <- int_knots[c(5:length(int_knots))]
ndx<-opt_num_knots - 4
dx<-(0.4 - 0)/ndx
out_knots <- seq(0-degree*dx,0.4+degree*dx,by=dx)
out_knots <- out_knots[1:(length(out_knots)-3)]
knots = c(out_knots, int_knots)
N_upd<-spline.des(knots,X,degree+1,0*X,outer.ok=T)$design
H_upd <- solve(t(N_upd)%*%N_upd)%*%t(N_upd)
beta_upd<-H_upd%*%Y

plot(X, f(X), type = "l", col = "blue", lwd = 2,
     main = "data", xlab = "X", ylab = "Y")

points(X, Y, col = "red", pch = 16, cex = 0.6)

lines(X, N_upd%*%beta_upd, col = "darkgreen", lwd = 2)

lines(X, N%*%beta, col = "yellow", lwd = 2)

legend("bottomright", legend = c("True f(X)", "Estimated f(X) with 4 knots in (0.4, 1]", "Previous one"
       col = c("blue", "darkgreen", "yellow", "red"), lty = c(2, 1, 1, NA), pch = c(NA, NA, NA, 16),
       lwd = c(2, 2, 2, NA), cex = 0.8)
```

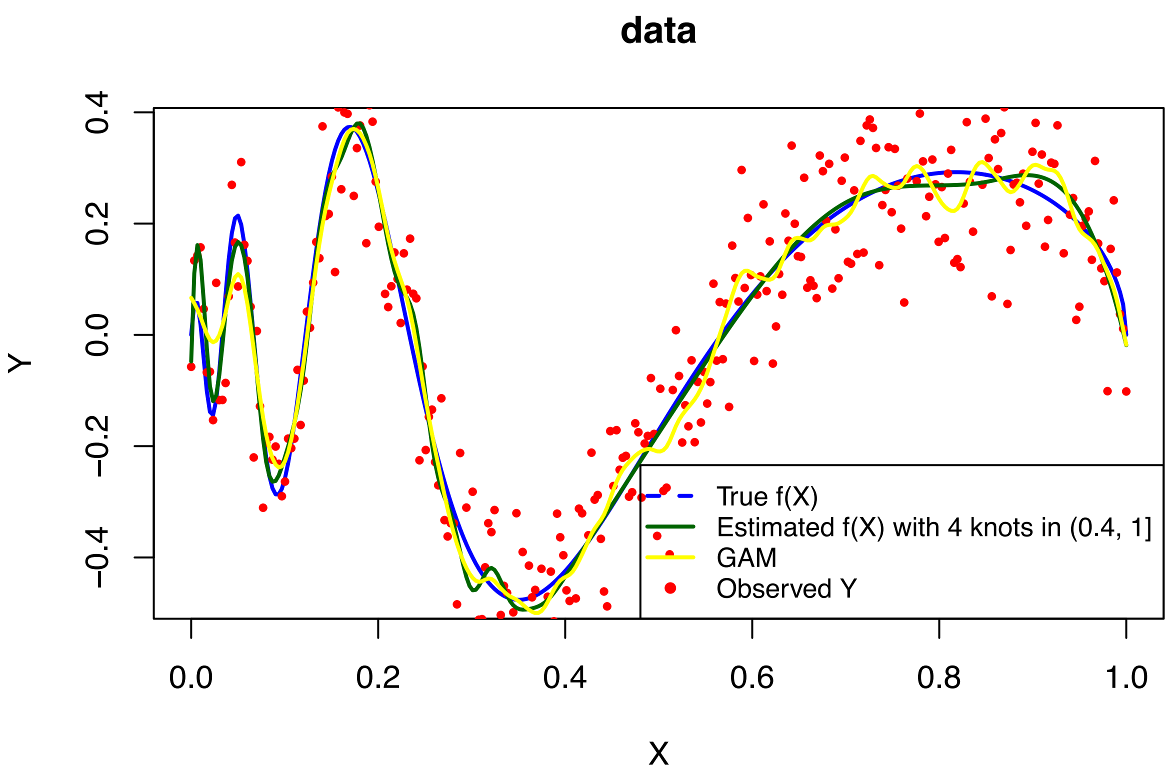


This time, model estimated both intervals much better, the only problem besides boundaries would be a boundary of those intervals. as we see on the plot, there is some fluctuation in point 0.3. This could be adjusted by changing algorithm of choosing knots distribution.

(c) Estimate f with penalized splines (use function `gam` of library `mgcv`, setting $m = 2$ and $k = 200$) and compare the result to the estimator from (b), putting both estimators on one plot. Comment on the results. (1 point)

```
degree <- 3

ndx<-4
dx<-(1 - 0.4)/ndx
int_knots<-seq(0.4-degree*dx,1+degree*dx,by=dx)
int_knots <- int_knots[c(5:length(int_knots))]
ndx<-opt_num_knots - 4
dx<-(0.4 - 0)/ndx
out_knots <- seq(0-degree*dx,0.4+degree*dx,by=dx)
out_knots <- out_knots[1:(length(out_knots)-3)]
knots = c(out_knots, int_knots)
N_upd<-spline.des(knots,X,degree+1,0*X,outer.ok=T)$design
H_upd <- solve(t(N_upd)%*%N_upd)%*%t(N_upd)
beta_upd<-H_upd%*%Y

GAM <- gam(Y~s(X, k = 200, m = 2))
Y_hat_gam <- predict(GAM)

plot(X, f(X), type = "l", col = "blue", lwd = 2,
     main = "data", xlab = "X", ylab = "Y")

points(X, Y, col = "red", pch = 16, cex = 0.6)
```

```
lines(X, N_upd%*%beta_upd, col = "darkgreen", lwd = 2)

lines(X, Y_hat_gam, col = "yellow", lwd = 2)

legend("bottomright", legend = c("True f(X)", "Estimated f(X) with 4 knots in (0.4, 1]", "GAM", "Observe
       col = c("blue", "darkgreen", "yellow", "red"), lty = c(2, 1, 1, NA), pch = c(NA, NA, NA, 16),
       lwd = c(2, 2, 2, NA), cex = 0.8)
```



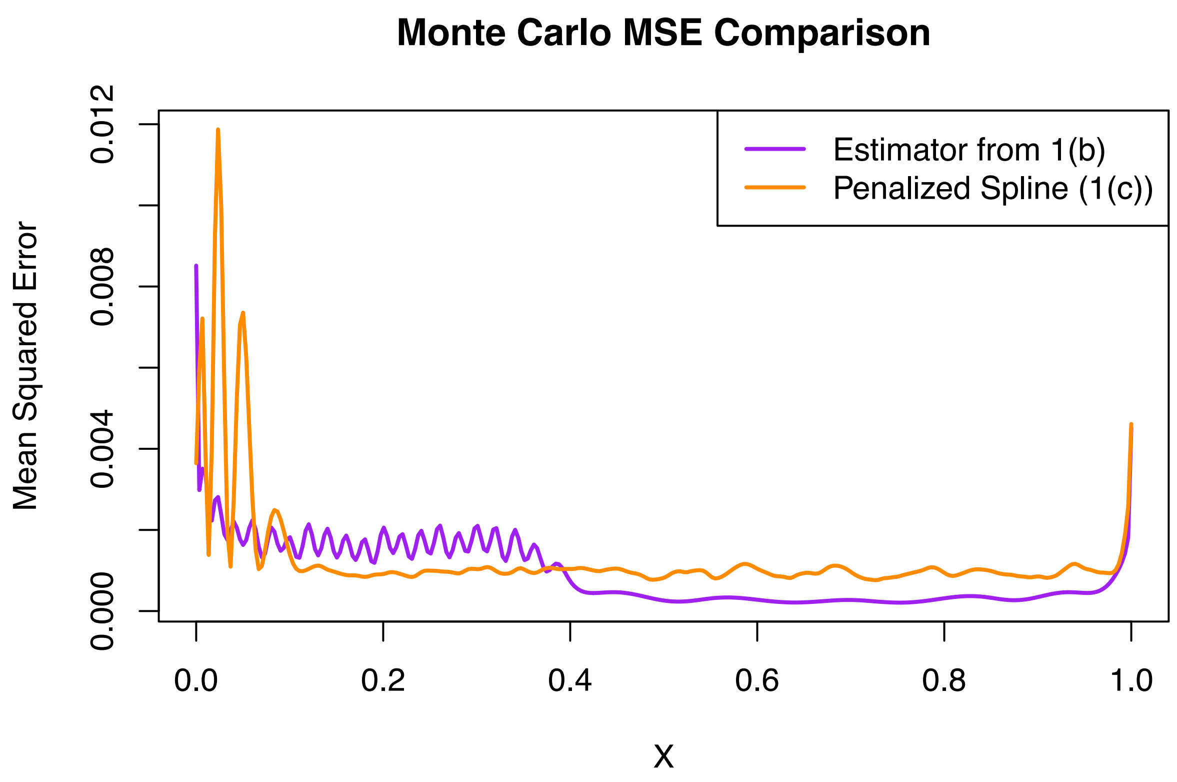
GAM also showed that it has higher variance in (0.4, 1) and lower bias at (0, 0.4). Maybe, changing parameters could help.

2. Simulate 300 samples as in 1, setting seed again to 2425. Obtain estimators for each sample as in 1(b) and 1(c). With this, get Monte Carlo estimators of the mean squared errors at each point x of these two estimators and plot them on one plot as a function of x, putting a legend. Comment on the results. (2 points)

```
num_simulations <- 300

mse_custom <- matrix(0, nrow = num_simulations, ncol = length(X))
mse_gam <- matrix(0, nrow = num_simulations, ncol = length(X))

ndx<-4
dx<-(1 - 0.4)/ndx
int_knots<-seq(0.4-degree*dx,1+degree*dx,by=dx)
int_knots <- int_knots[c(5:length(int_knots))]
ndx<-opt_num_knots - 4
dx<-(0.4 - 0)/ndx
out_knots <- seq(0-degree*dx,0.4+degree*dx,by=dx)
out_knots <- out_knots[1:(length(out_knots)-3)]
knots = c(out_knots, int_knots)
```

```r
for (i in 1:num_simulations) {
  break
  epsilon <- rnorm(length(X), mean = 0, sd = 0.1)
  Y_sim <- f(X) + epsilon
  N_upd<-spline.des(knots,X,degree+1,0*X,outer.ok=T)$design
  H_upd <- solve(t(N_upd)%*%N_upd)%*%t(N_upd)
  beta_upd<-H_upd%*%Y_sim
  Y_hat_custom <- N_upd %*% beta_upd

  gam_model <- gam(Y_sim ~ s(X, k = 200, m = 2))
  Y_hat_gam <- predict(gam_model)

  mse_custom[i, ] <- (Y_hat_custom - f(X))^2
  mse_gam[i, ] <- (Y_hat_gam - f(X))^2

  cat("\rFinished", i, "of", num_simulations)
}



mean_mse_custom <- colMeans(mse_custom)
mean_mse_gam <- colMeans(mse_gam)

#save(mean_mse_custom, file = "mean_mse_custom.Rdata")
#save(mean_mse_gam, file = "mean_mse_gam.Rdata")
```

```r
load("mean_mse_custom.Rdata")
load("mean_mse_gam.Rdata")

plot(X, mean_mse_custom, type = "l", col = "purple", lwd = 2, ylim = range(c(mean_mse_custom, mean_mse_
     xlab = "X", ylab = "Mean Squared Error", main = "Monte Carlo MSE Comparison")
lines(X, mean_mse_gam, col = "darkorange", lwd = 2)
legend("topright", legend = c("Estimator from 1(b)", "Penalized Spline (1(c))"),
       col = c("purple", "darkorange"), lty = 1, lwd = 2)
```



Again: higher variance for 1b estimator on (0, 0.4) as t has more knots there, but low bias and variance on (0.4, 1) as there are only 4 knots there. For 1c variance is consistent almost everywhere, except (0, 0.1) and boundaries.

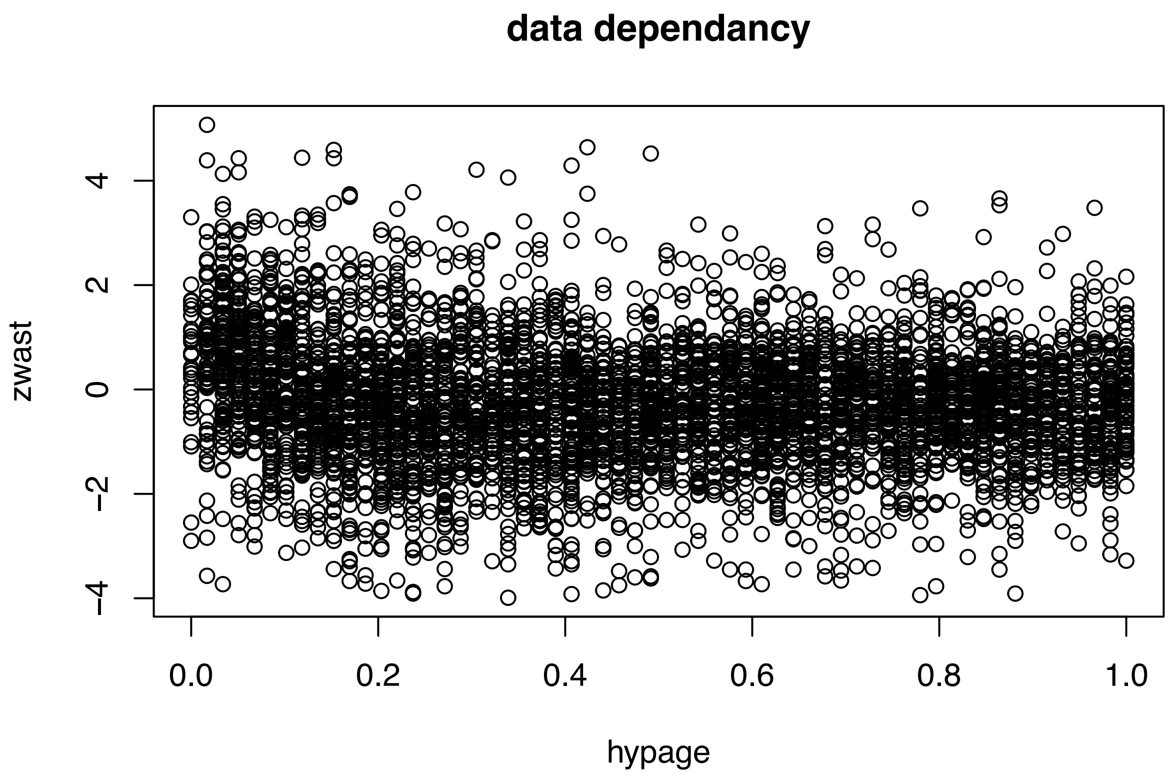
Problem 3

Read the dataset `Kenya DHS` into `R` and consider variables `zwast` as a response and `hypage` as a covariate. Scale the covariate to the interval $[0, 1]$ and estimate f in the model

$$\texttt{zwast}_i = f(\texttt{hypage}_i) + \varepsilon_i, \quad i = 1, \ldots, 4686,$$

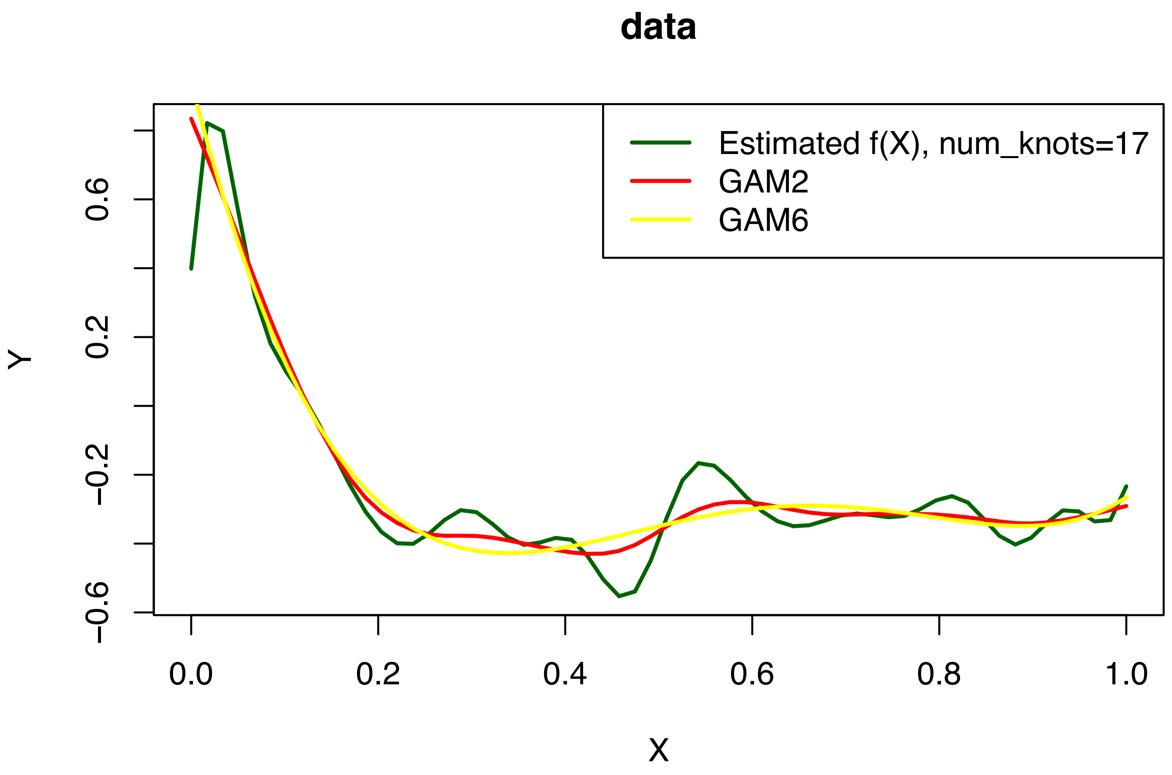
with a least-squares spline estimator of degree 3 and equidistant knots. Choose the number of equidistant knots by GCV. Plot the resulting estimator without the data. Next, estimate f with penalized splines, using function `gam` of library `mgcv` and setting $m = 2, k = 40$. Add this estimator to the previous plot and comment on the results. Refit the penalized spline estimator setting $m = 6$ now. Add the resulting estimator to the previous plot and comment on the results. (5 points)

```r
data=read.table("KenyaDHS.txt",header=TRUE)
attach(data)

clrs = c("red", "green", "yellow")

y <- data[order(data$hypage),]$zwast
x <- data[order(data$hypage),]$hypage
x <- ((x- min(x)) /(max(x)-min(x))) # normalizing


plot(x, y, main="data dependancy", xlab = "hypage", ylab="zwast")
```



```r
GCV <- function(X, Y, degree) {
  gcv <- function(ndx, X, Y, degree) {
    dx <- 1 / ndx
    knots <- seq(0 - degree * dx, 1 + degree * dx, by = dx)
    N <- spline.des(knots, X, degree + 1, rep(0, length(X)), outer.ok = TRUE)$design
    H <- solve(t(N) %*% N) %*% t(N)
    beta <- H %*% Y
    Y_hat <- N %*% beta
    residuals <- Y - Y_hat
    return(sum(residuals^2)/(1-sum(diag(H)/length(Y)))^2)
  }
  return(optimize(function(ndx) gcv(round(ndx),X=X,Y=Y,degree=degree),interval=c(2, 20))$minimum)
}

degree=3

opt_num_knots <- GCV(x, y, degree)
opt_num_knots <- round(opt_num_knots)

ndx <- opt_num_knots
degree <- 3
dx <- 1/ndx
knots <- seq(0-degree*dx,1+degree*dx,by=dx)
N=spline.des(knots,x,degree+1,0*x,outer.ok=T)$design
H  <-  solve(t(N)%*%N)%*%t(N)
beta <- H%*%y

gam_model_2 <- gam(y ~ s(x, k = 40, m = 2))
Y_hat_gam_2 <- predict(gam_model_2)
```

```r
gam_model_6 <- gam(y ~ s(x, k = 40, m = 6))
Y_hat_gam_6 <- predict(gam_model_6)


plot(x, N%*%beta, type = "l", col = "darkgreen", lwd = 2,
     main = "data", xlab = "X", ylab = "Y")

lines(x, Y_hat_gam_2, col = "red", lwd = 2)

lines(x, Y_hat_gam_6, col = "yellow", lwd = 2)

legend("topright",
       legend = c(paste0("Estimated f(X), num_knots=",opt_num_knots), "GAM2", "GAM6"),
       col = c("darkgreen", "red", "yellow"), lty = 1, lwd = 2
       )
```



GAM m parameter penalizes model to be more smooth with higher m, as we see here, GAM6 is smoother. Least square spline seem to have higher variance, as it doesnt have complexity penalty like GAM.