Modele regresji i ich zastosowania - raport 5.

Patryk Krukowski(249824)

7 czerwca 2021

Spis treści

1 Regressogram i lokalne średnie

1.1 Wstęp

W tej sekcji zajmiemy się skonstruowaniem regressogramu oraz lokalnych średnich, będacych nieparametrycznymi estymatorami funkcji regresji $r(\cdot)$. W związku z tym niech:

- $r(x) = 10sin(2\pi x)$
- $n = 500$,
- $\sigma = 0.5$,
- $\boldsymbol{x} = \left(\frac{1}{500}, \frac{2}{500}, \ldots, 1\right)$

Przechodzimy teraz do wykonania zadań laboratoryjnych.

1.2 Zadanie nr 1

Generujemy n obserwacji $Y_1, Y_2, \ldots, Y_n$ postaci

$$Y_i = r(x_i) + \sigma\epsilon_i, i = 1, 2, \ldots, n,$$

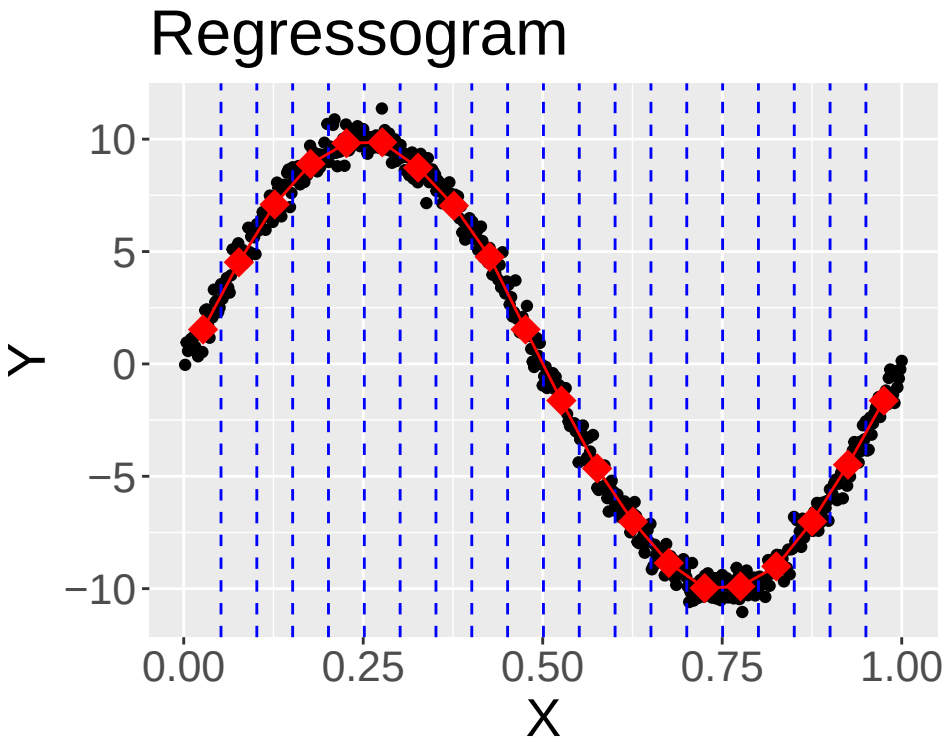
gdzie $\epsilon_1, \ldots, \epsilon_n$ i.i.d. $\mathcal{N}(0, 1)$.

```
n <- 500
sigma <- 0.5
x <- seq(1/n, 1, 1/n)

#Generujemy wektor Y i r(x)
r <- rep(0, n)
for (i in 1:n) {
  r[i] <- 10*sin(2*pi*x[i])
}
y <- r+rnorm(n, mean=0, sd=sigma)
```

1.3 Zadanie nr 2

Konstruujemy regressogram, dzieląc przedział [0,1] na $m = 20$ przedziałów równej długości $h = \frac{1}{m}$. Aby wykonać to zadanie, możemy użyć funkcji *regressogram* z biblioteki *HoRM*. My jednak jej **nie** użyjemy. Powód? W otrzymanym wykresie można łatwo dokonać błędnej interpretacji. Spójrzmy na to, jak wygląda.

```
p1 <- regressogram(x,y,nbins=20, main='Regressogram')
p1
```



Rysunek 1: Regressogram utworzony dla syntetycznie wygenerowanych danych

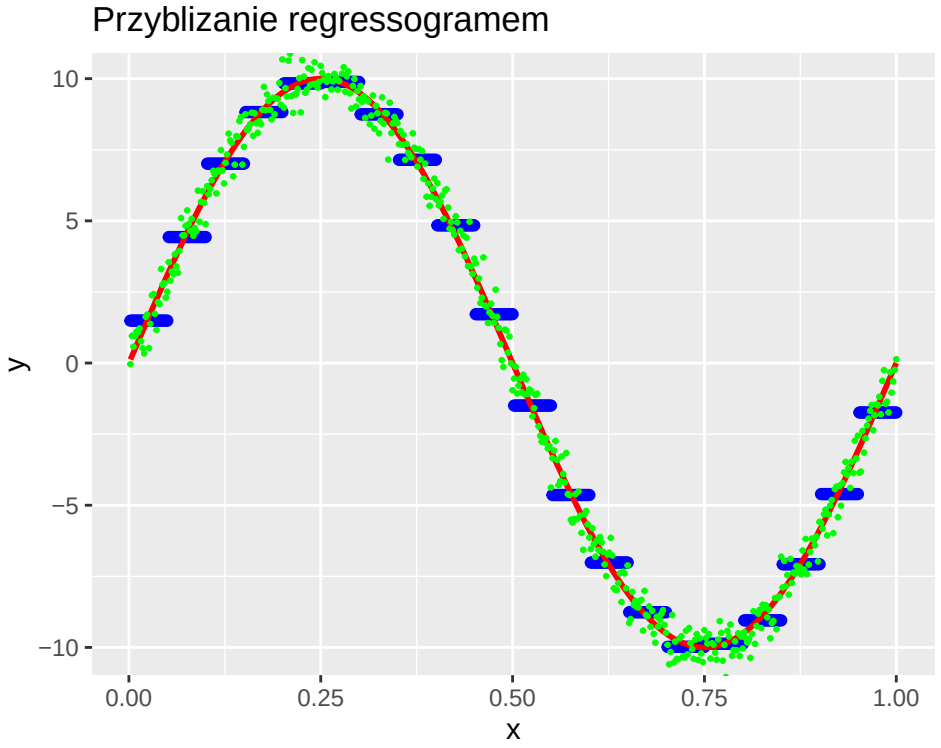
Wartości, które powinny być stałe na otrzymanych przedziałach wyglądają jak pojedyncze punkty, ale bardziej pogrubione. W związku z tym sami zaimplementujemy ten estymator w poniższym kodzie.

```
regres <- function(x,y,k){
  # k = liczba przedzialów
  B = seq(0,1,length=k+1)

  #znajdujemy przedzial, w którym znaduje sie konkretna obserwacja
  WhichBin = findInterval(x,B)

  #"podstawiamy do wzoru"
  N = tabulate(WhichBin)
  m.hat = rep(0,k)
  for(j in 1:k){
    if(N[j]>0) m.hat[j] = mean(y[WhichBin == j])
  }
  return(list(bins=B,m.hat=m.hat))
}
```

1.4 Zadanie nr 3

Rysujemy wykres, o który poproszono nas w zadaniu.

```r
regres_20 <- regres(x,y,20)

#tworzymy ramke danych do ggplot
df_20 <- data.frame(x, y, r, rep(0,n))
colnames(df_20)[4] <- 'reg_punkty'

#przyporządkowujemy dla x odpowiednie wartości regressogramu
for (j in 1:n) {
  for (i in 1:20) {
    if (p1$data$z[j] == i) {
    df_20$reg_punkty[j] <- regres_20$m.hat[i]
    }
  }
}

#rysujemy
ggplot(df_20, aes(x=x, y=reg_punkty)) +
  geom_point(color='blue') +
  geom_line(y=r, color='red', lwd=1) +
  geom_point(y=y, color='green', size=0.5) +
  labs(x='x', y='y') +
  ggtitle('Przybli■anie regressogramem')
```



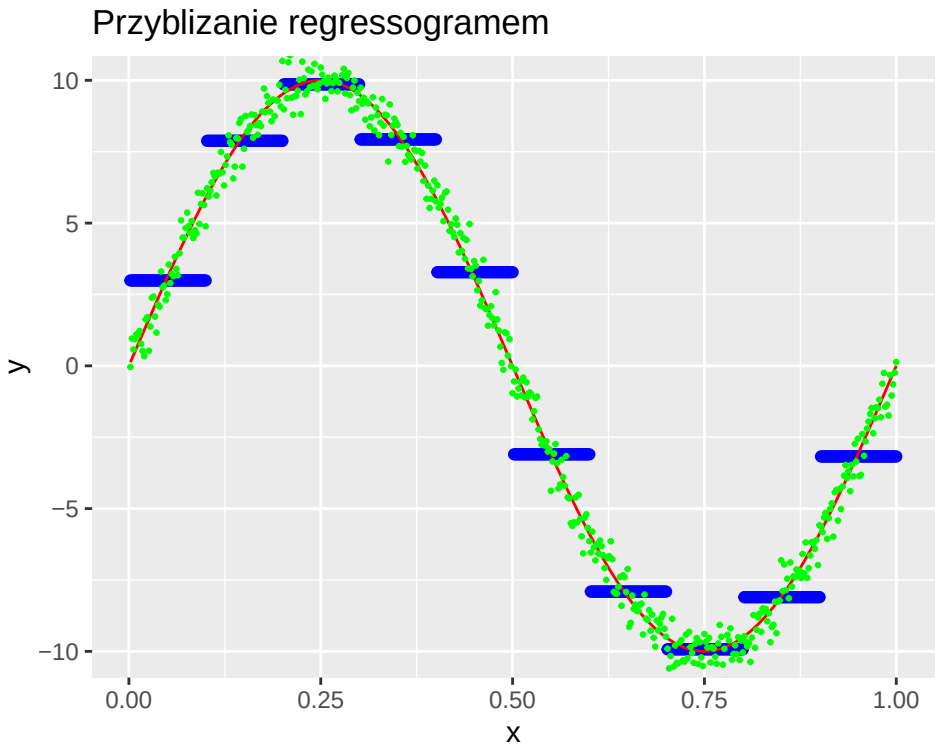
Rysunek 2: Estymacja funkcji regresji regressogramem dla h=0.05; regressogram - kolor niebieski, estymowana funkcja - kolor czerwony, obserwowane wartości y - kolor zielony

1.5 Zadanie nr 4

Powtarzamy analizę z dwóch poprzednich zadań dla

- $h = \frac{1}{10}$.

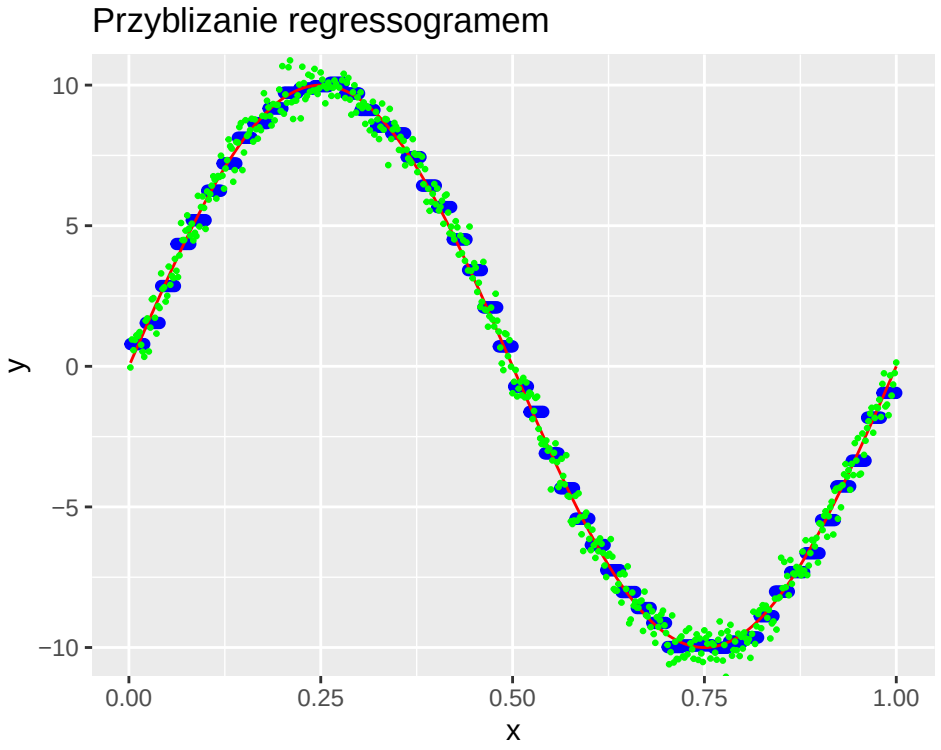
```
#m=10
p2 <- regressogram(x,y,nbins=10)
regres_10 <- regres(x,y,10)
df_10 <- data.frame(x, y, r, rep(0,n))
colnames(df_10)[4] <- 'reg_punkty'
for (j in 1:n) {
  for (i in 1:10) {
    if (p2$data$z[j] == i) {
      df_10$reg_punkty[j] <- regres_10$m.hat[i]
    }
  }
}

ggplot(df_10, aes(x=x, y=reg_punkty)) +
  geom_point(color='blue') +
  geom_line(y=r, color='red') +
  geom_point(y=y, color='green', size=0.5) +
  labs(x='x', y='y') +
  ggtitle('Przybli■anie regressogramem')
```



Rysunek 3: Estymacja funkcji regresji regressogramem dla h = 0.01; regressogram - kolor niebieski, estymowana funkcja - kolor czerwony, obserwowane wartości y - kolor zielony

- $h = \frac{1}{50}$.

```
#m=50
p3 <- regressogram(x,y,nbins=50)
regres_50 <- regres(x,y,50)
df_50 <- data.frame(x, y, r, rep(0,n))
colnames(df_50)[4] <- 'reg_punkty'
for (j in 1:n) {
  for (i in 1:50) {
    if (p3$data$z[j] == i) {
      df_50$reg_punkty[j] <- regres_50$m.hat[i]
    }
  }
}

ggplot(df_50, aes(x=x, y=reg_punkty)) +
  geom_point(color='blue') +
  geom_line(y=r, color='red') +
  geom_point(y=y, color='green', size=0.5) +
  labs(x='x', y='y') +
  ggtitle('Przybli■anie regressogramem')
```



Rysunek 4: Estymacja funkcji regresji regressogramem dla h = 0.02; regressogram - kolor niebieski, estymowana funkcja - kolor czerwony, obserwowane wartości y - kolor zielony

Analizując rysunki nr 2, 3 i 4, dochodzimy do następującego wniosku.

Wniosek 1 *Im liczba przedziałów użytych do konstrukcji regressogramu jest większa, tym estymator ten lepiej przybliża funkcję regresji. Oczywiście, o ile badana funkcja jest "dostatecznie regularna".*

1.6 Zadanie nr 5

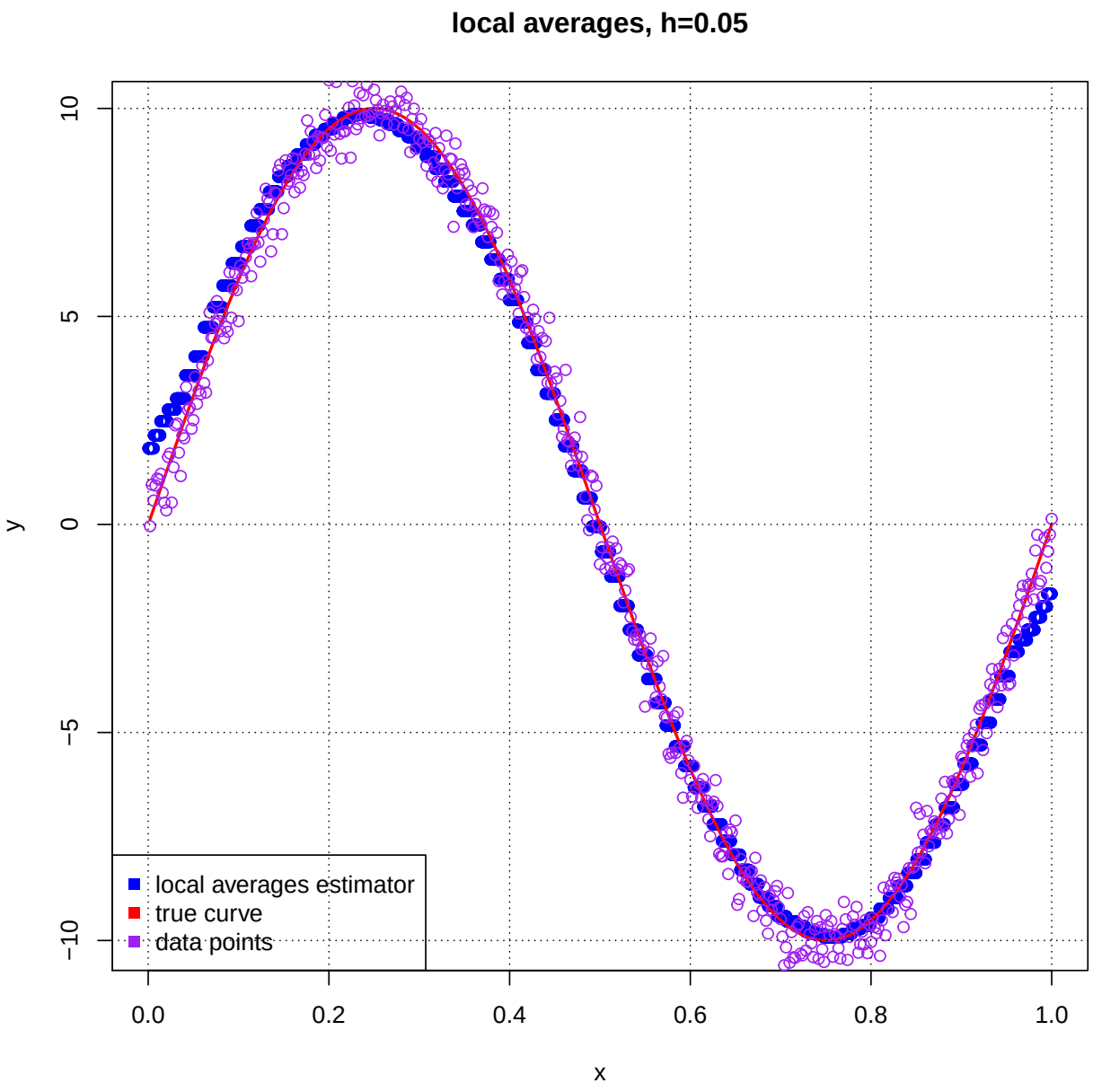
Teraz powtórzymy zadania 3-4, gdy estymatorem funkcji regresji jest estymator lokalnych średnich. Najpierw implementacja.

```
local_avg <- function(x,y,h) {
  #pusty wektor na wartości estymatora
  Y <- c()

  #iterujemy sie po pewnej gestej siatce punktów, przesuwając "okno"
  for (x_star in seq(0.01, 1, by=0.01)) {

    #inicjalizacja wektora wag
    wt <- rep(0, n)

    #przedzial, do którego wpadają kolejne obserwacje
    window <- c(max(x_star - h,0), min(x_star + h,1))

    #iterujemy sie po x, sprawdzając, które obserwacje wpadly do danego przedzia-
lu
    for (x_point in x) {
      if (x_point <= window[2] & x_point > window[1]) {
        idx <- which(x_point == x)
        wt[idx] <- 1
        }
    }
    #trzeba uwa■ac - sum(wt) mo■e byc równe 0
    if (sum(wt) > 0) {
      y_star <- sum(y*wt)/sum(wt)
      Y <- append(Y, rep(y_star, sum(wt)))
    } else {
      y_star <- 0
    }
  }
  #zwracamy data frame z wynikami
  z <- seq(1/length(Y),1,by=1/length(Y))
  results <- data.frame(z, Y)
  return(results)
}
```

Niech

- $h = 0.05 (= \frac{1}{20})$.

```r
loc_avg_0.05 <- local_avg(x,y,1/20)
plot(loc_avg_0.05, main='local averages, h=0.05', col='blue', xlab='x', ylab='y')
lines(x, 10*sin(2*pi*x), lwd=2, col='red')
points(x,y,pch=1, col='purple')
grid(col='black')
legend('bottomleft',
       legend=c("local averages estimator", "true curve", "data points"),
       col=c('blue', 'red', 'purple'),
       pch = 15
       )
```

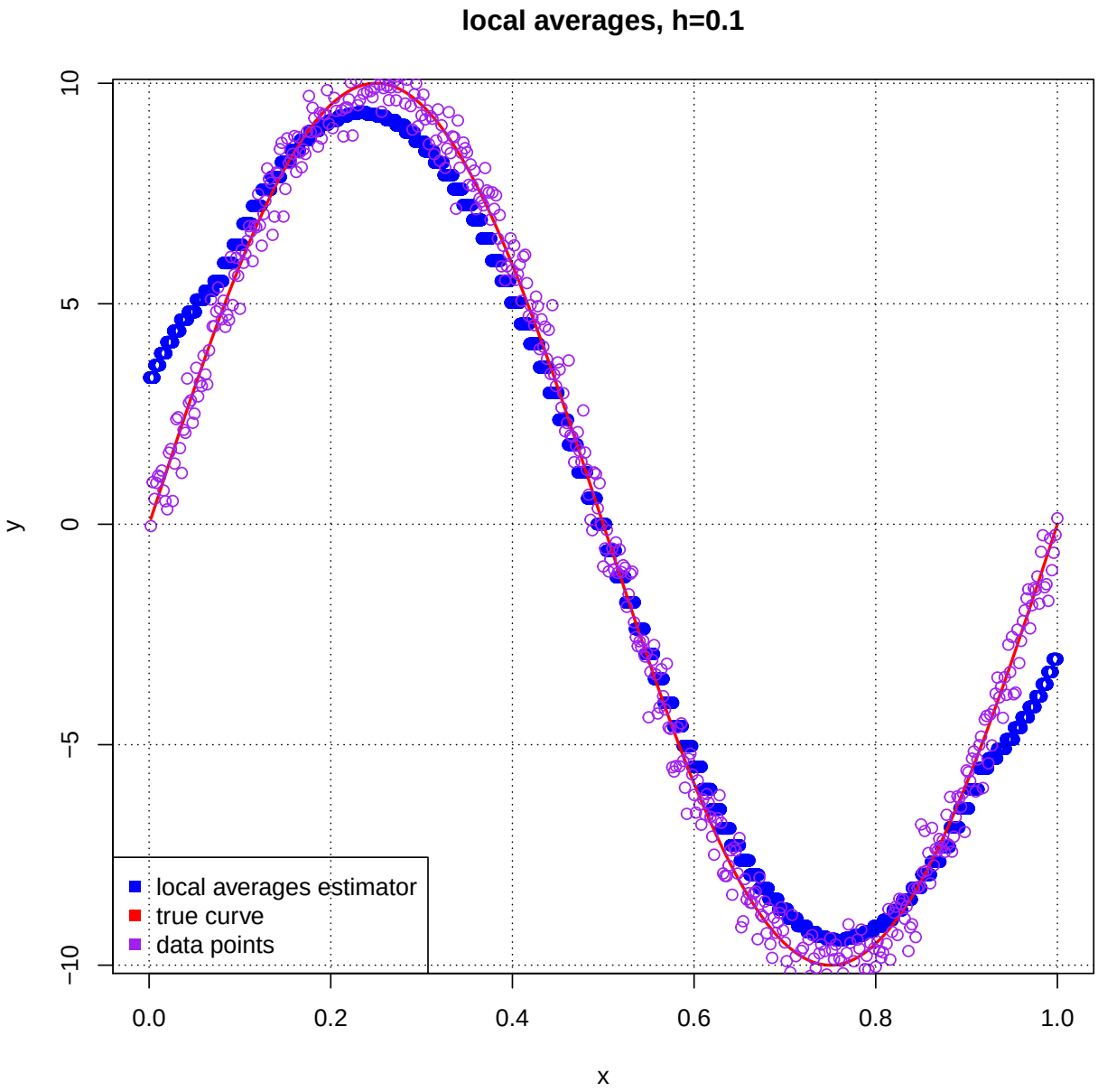


Rysunek 5: Estymacja funkcji regresji estymatorem lokalnych średnich dla h = 0.05

Estymator ten radzi sobie bardzo dobrze z przybliżeniem funkcji regresji, choć możemy zaobserwować lekkie problemy z estymacją funkcji regresji na brzegach. Zobaczmy, jak poradzi sobie ten estymator, gdy zmienimy parametr h.

- $h = 0.1$ - z rysunku 6 wynika, że zwiększenie parametru h doprowadziło do gorszej estymacji i obserwujemy jeszcze większe kłopoty z estymacją na brzegach.

```
#h = 1/10
loc_avg_0.1 <- local_avg(x,y,1/10)
plot(loc_avg_0.1, main='local averages, h=0.1', col='blue', xlab='x', ylab='y')
lines(x, 10*sin(2*pi*x), lwd=2, col='red')
points(x,y,pch=1, col='purple')
grid(col='black')
legend('bottomleft',
       legend=c("local averages estimator", "true curve", "data points"),
       col=c('blue', 'red', 'purple'),
       pch = 15
)
```

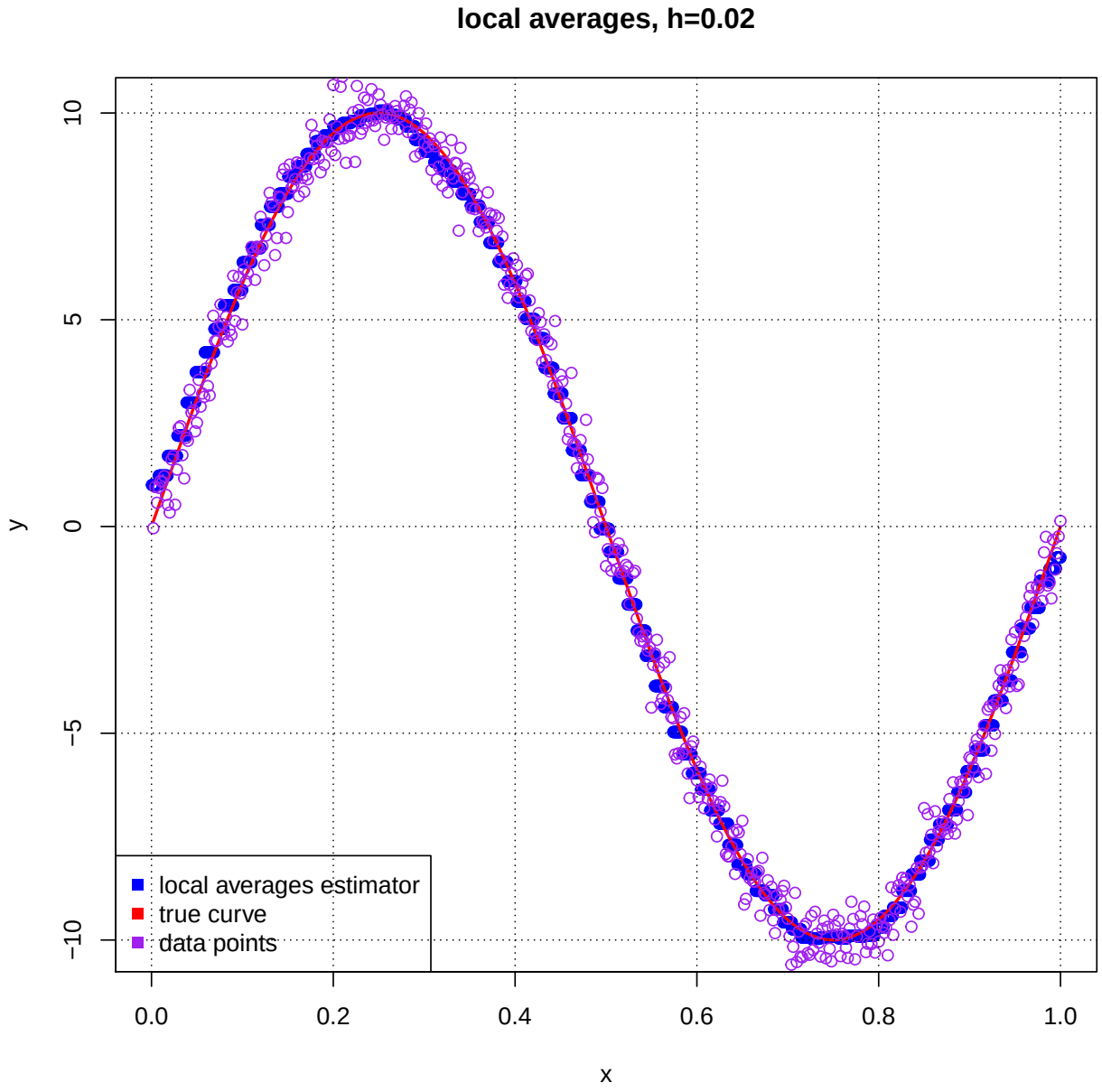


Rysunek 6: Estymacja funkcji regresji estymatorem lokalnych średnich dla h = 0.1

- $h = 0.02 \quad (= \frac{1}{50})$ - rysunek nr 7 mówi nam o tym, że zmniejszenie parametru h prowadzi do dokładniejszej estymacji (jest dobrze nawet dla brzegów!).

```
#h = 1/50
loc_avg_0.02 <- local_avg(x,y,1/50)
plot(loc_avg_0.02, main='local averages, h=0.02', col='blue', xlab='x', ylab='y')
lines(x, 10*sin(2*pi*x), lwd=2, col='red')
points(x,y,pch=1, col='purple')
grid(col='black')
legend('bottomleft',
       legend=c("local averages estimator", "true curve", "data points"),
```

```
        col=c('blue', 'red', 'purple'),
        pch = 15
)
```



Rysunek 7: Estymacja funkcji regresji estymatorem lokalnych średnich dla h = 0.02

Wniosek 2 *Zmniejszenie parametru h prowadzi do dokładniejszej estymacji. Stąd parametr h możemy interpretować jako parametr wygładzający.*

Wniosek 3 *Dla nieodpowiednio dobranego parametru h estymator lokalnych średnich nie radzi sobie z estymacją krzywej na brzegach.*

Ciekawym zagadnieniem wydaje się uwzględnienie wektora parametrów wygładzających zamiast rzeczywistej dodatniej liczby, ale nie będziemy się na tym zatrzymywać. Natomiast warto zwrócić uwagę na to, że przybliżamy funkcję **ciągłą** funkcją **nieciągłą**, co wydaje się być dość kontrowersyjne.

2 Estymator Nadaraya-Watsona i wielomiany lokalne

2.1 Wstęp

W tej sekcji zajmiemy się skonstruowaniem estymatora Nadaraya-Watsona oraz wielomianów lokalnych, będacych także nieparametrycznymi estymatorami funkcji regresji $r(\cdot)$, ale ciągłymi. Tym razem niech:

- $$r(x) = \sqrt{x(1-x)}\sin\left(\frac{2.1\pi}{x+0.05}\right), \quad 0 \leq x \leq 1,$$
- $n = 1000$
- $\sigma = 1$
- $\boldsymbol{x} = \left(\frac{1}{1000}, \frac{2}{1000}, \ldots, 1\right)$

Zauważmy, że tym razem estymować będziemy funkcję o wiele bardziej nieregularną, niż w poprzednim zadaniu. Funkcja ta jest tzw. funkcją Dopplera.

2.2 Zadanie nr 1

Generujemy n obserwacji $Y_1, Y_2, \ldots, Y_n$ postaci

$$Y_i = r(x_i) + \sigma\epsilon_i, i = 1, 2, \ldots, n,$$

gdzie $\epsilon_1, \ldots, \epsilon_n$ i.i.d. $\mathcal{N}(0,1)$.

```
n_1 <- 1000
x_1 <- seq(1/n_1, 1, 1/n_1)
sigma_1 <- 1
r_1 <- c(0, n_1)
for (k in 1:n_1) {
  r_1[k] <- sqrt(x_1[k]*(1-x_1[k]))*sin(2.*pi/(x_1[k]+0.05))
}
y_1 <- r_1+rnorm(n_1, 0, sigma_1)
```

2.3 Zadanie nr 2

Skonstruujemy teraz estymator jądrowy Nadaraya-Watsona $\hat{r}_n(\cdot)$ rozważanej funkcji regresji $r(\cdot)$, oparty na jądrze gaussowskim $K(x) = \frac{1}{2\pi}e^{-\frac{x^2}{2}}$, dobierając parametr h za pomocą metody *leave-one-out cross-validation*. Najpierw implementujemy funkcje *gaussian_kernel*, *nw_estimator*, *LOOCV* zwracające odpowiednio wektor wartości funkcji $K(\cdot)$, wektor wartości estymatora jądrowego N-W oraz wartość *leave-one-out cross-validation*.

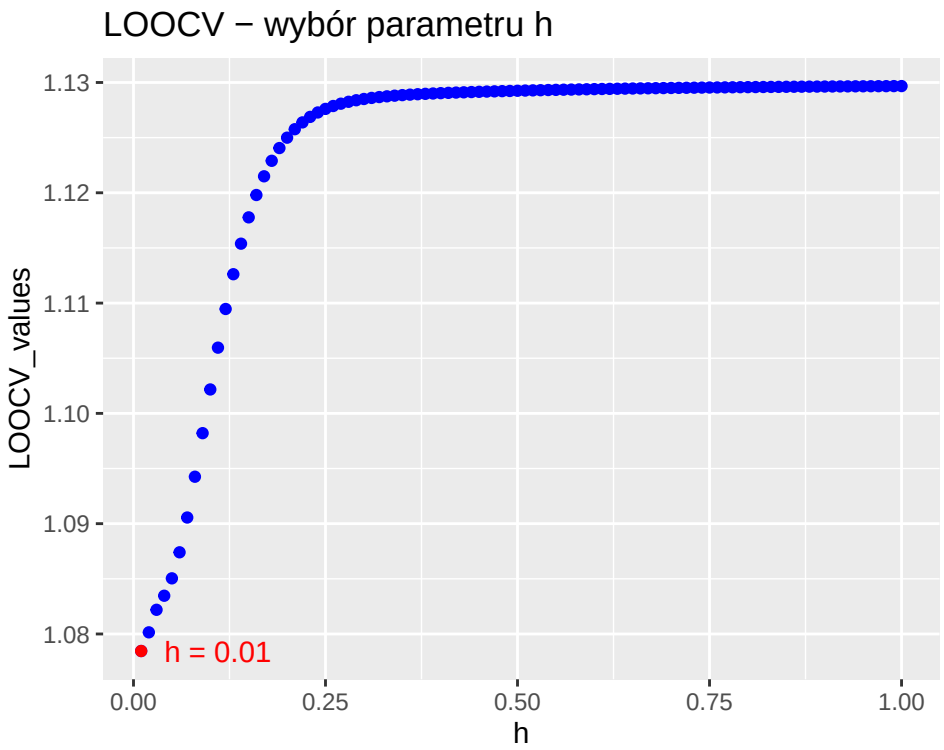
```
gaussian_kernel <- function(x) {
  return(dnorm(x, 0, 1))
}

nw_estimator <- function(x, y, h) {
  Kij <- outer(x, x, function(x, x_i) gaussian_kernel((x-x_i)/h))
  S <- Kij / rowSums(Kij)
  return(S%*%y)
}

LOOCV <- function(h) {
  Kij <- outer(x_1, x_1, function(x, x_i) gaussian_kernel((x-x_i)/h))
  S <- Kij / rowSums(Kij)
  mean(((y_1-S%*%y_1) / (1-diag(S)))^2)
}
```

Dobieramy teraz h metodą wspomnianą wyżej. Wybór ten zawężimy do gęstej siatki punktów postaci $\frac{i}{n}, \quad i = 1, 2, \ldots, n$.

```
h <- seq(0.01, 1, by=0.01)
LOOCV_values <- sapply(h, LOOCV)
h_opt <- h[which.min(LOOCV_values)]
h_opt

## [1] 0.01
```

Optymalna wartość parametru $\hat{h}_n$ to 0.01. Zwizualizujmy jeszcze ten wybór, rysując wykres funkcji $\hat{R}_{CV}(h)$.

```
ggplot(data=data.frame(h, LOOCV_values), aes(x=h, y=LOOCV_values)) +
  geom_point(colour='blue') +
  geom_point(data=data.frame(h, LOOCV_values)[h == h_opt,],
             aes(x=h, y=LOOCV_values),
             colour = 'red'
             ) +
  geom_text(data=data.frame(h, LOOCV_values)[h == h_opt,],
            label="h = 0.01",
            aes(x=h+0.1, y=LOOCV_values),
            colour='red'
            ) +
  ggtitle('LOOCV - wybór parametru h')
```



Rysunek 8: Wybór parametru h w oparciu o metodę LOOCV dla estymatora N-W dla sigma=1

A zatem optymalna wartość parametru została dobrana we właściwy sposób.

2.4 Zadanie nr 3

Nasz cel jest taki sam, jak w poprzednim zadaniu z dokładnością do tego, że tym razem nieparametrycznym estymatorem funkcji $r(\cdot)$ jest estymator $\tilde{r}(\cdot)$ oparty na wielomianach lokalnych pierwszego stopnia.

Do implementacji estymatora wielomianów lokalnych stopnia pierwszego użyjemy funkcji *locpoly* z paczki *KernSmooth*. Natomiast do wyboru optymalnego parametru h użyjemy funkcji *np.cv* z biblioteki *PLRModels*.

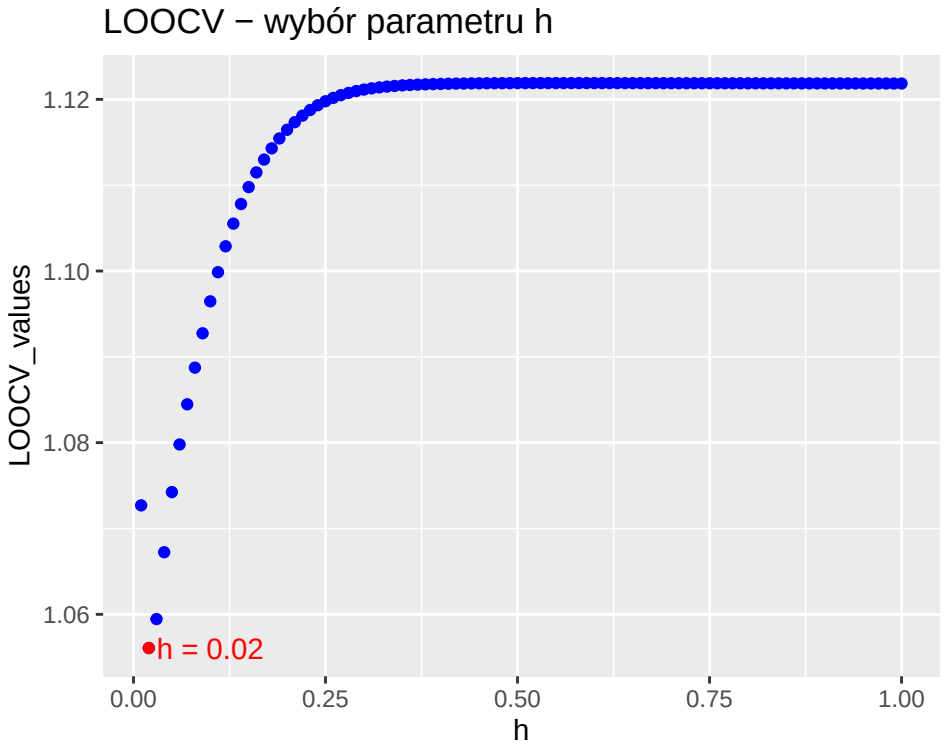
```
loc_pol <- function(x,y,h) {
  return(locpoly(x,y, kernel='normal', bandwidth = h, degree=1,
                 range.x = c(min(x), max(x))))
}

LOOCV_fit <- np.cv(data=matrix(c(y_1, x_1), ncol=2, nrow=n_1),
                   h.seq=h,
                   w=c(0,1),
                   estimator='LLP',
                   kernel='gaussian'
                  )
h_opt_1 <- LOOCV_fit$h.opt[2,1]
h_opt_1

## [1] 0.02
```

Uwaga 1 *W funkcji locpoly dobraliśmy wagi jako jądra gaussowskie, ponieważ takie wagi mają tę własność, że najbliższe punkty z sąsiedztwa badanego aktualnie punktu mają większe znaczenie w stosunku do punktów bardziej oddalonych.*

Uwaga 2 *W funkcji np.cv ustawiamy na nasze potrzeby estimator='LLP' (LLP = Local Linear Polynomials). Domyślnie, w funkcji tej mamy* num.ln = 1. *Oznacza to, że procedura ta realizuje leave-one-out cross-validation (o czym można przeczytać w dokumentacji).*

Podobobnie, jak poprzednio, sprawdzimy, czy liczba $\hat{h}_n$ została dobrana we właściwy sposób.

```
df_1 <- data.frame(h, LOOCV_fit$CV)
colnames(df_1)[2] <- 'LOOCV_values'
ggplot(data=df_1, aes(x=h, y=LOOCV_values)) +
  geom_point(colour='blue') +
  geom_point(data=df_1[h == h_opt_1,],
             aes(x=h, y=LOOCV_values),
             colour = 'red'
  ) +
  geom_text(data=df_1[h == h_opt_1,],
            label="h = 0.02",
            aes(x=h*5, y=LOOCV_values),
            colour='red'
  ) +
  ggtitle('LOOCV - wybór parametru h')
```



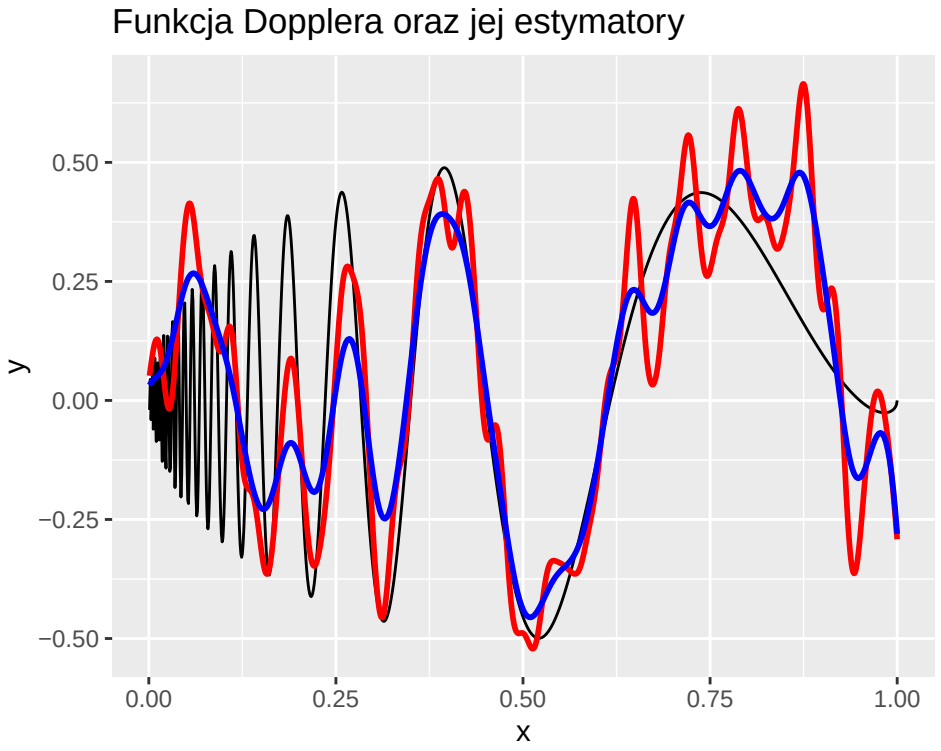
Rysunek 9: Wybór parametru h w oparciu o metodę LOOCV dla estymatora lokalnych wielomianów stopnia pierwszego dla sigma=1

Jak możemy odczytać z powyższego rysunku, liczba $\hat{h}_n = 0.02$ została dobrana we właściwy sposób.

2.5 Zadanie nr 4

Teraz narysujemy wykres funkcji Dopplera wraz z jej badanymi w tej sekcji estymatorami, skonstruowanymi w oparciu o wyliczone w szybszym zadaniu optymalne parametry h.

```
df_2 <- data.frame(x_1, r_1, nw_estimator(x_1, y_1, h_opt))
df_3 <- data.frame(loc_pol(x_1, y_1, h_opt_1)$x,
                   loc_pol(x_1, y_1, h_opt_1)$y
                   )
colnames(df_2)[3] <- c('NW_estimator')
colnames(df_3)[1] <- 'x'
colnames(df_3)[2] <- 'local_polynomials_estimator'
ggplot(data=df_2, aes(x=x_1, y=r_1)) +
  geom_line() +
  geom_line(aes(x= x_1, y=NW_estimator), colour='red', lwd=1) +
  geom_line(data=df_3, aes(x=x, y=local_polynomials_estimator), colour='blue', lwd=1) +
  ggtitle('Funkcja Dopplera oraz jej estymatory') +
  labs(x='x', y='y')
```



Rysunek 10: Funkcja Dopplera (kolor czarny) oraz jej estymatory N-W oraz wielomianów lokalnych stopnia pierwszego (odpowiednio kolor czerwony i niebieski) dla sigma=1

Estymatory nie radzą sobie zbyt dobrze, pomimo dobrania optymalnych h (szczególnie w pobliżu zera). Jest to spowodowane nie tylko trudną do estymacji funkcją, ale i dużym szumem σ (o tym przekonamy się jeszcze pod koniec tej sekcji).

2.6 Zadanie nr 5

Powtórzymy teraz analizę z sekcji nr 2 dla przypadku, gdy

(a) $\sigma = 0.5$

- Generujemy obserwacje $Y_i, \quad i = 1, \ldots, n$ z nowym parametrem σ.

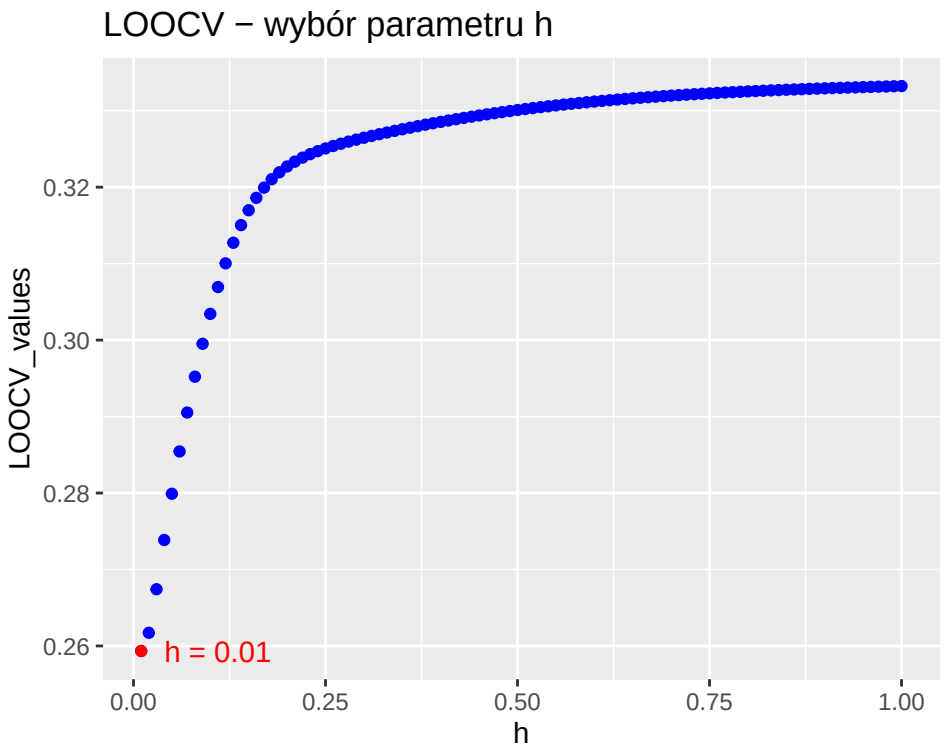
```
sigma_2 <- 0.5
y_2 <- r_1+rnorm(n_1, 0, sigma_2)
```

- Konstruujemy estymator Nadaraya-Watsona funkcji regresji oparty na jądrze gaussowskim, wyznaczając h przy pomocy LOOCV (znów użyjemy dla prostoty funkcji *np.cv*, zmieniając ustawienie *estimator='NW'*).

```
LOOCV_fit_nw_0.5 <- np.cv(data=matrix(c(y_2, x_1), ncol=2, nrow=n_1),
      h.seq=h,
      w=c(0,1),
      estimator='NW',
      kernel='gaussian'
)
h_opt_2 <- LOOCV_fit_nw_0.5$h.opt[2,1]
h_opt_2

## [1] 0.01
```

Sporządzamy także wykres estymatora $\hat{R}_{CV}(h)$.

```
df_3 <- data.frame(h, LOOCV_fit_nw_0.5$CV)
colnames(df_3)[2] <- 'LOOCV_values'
ggplot(data=df_3, aes(x=h, y=LOOCV_values)) +
  geom_point(colour='blue') +
  geom_point(data=df_3[h == h_opt_2,],
             aes(x=h, y=LOOCV_values),
             colour = 'red'
  ) +
  geom_text(data=df_3[h == h_opt_2,],
            label="h = 0.01",
            aes(x=h+0.1, y=LOOCV_values),
            colour='red'
  ) +
  ggtitle('LOOCV - wybór parametru h')
```



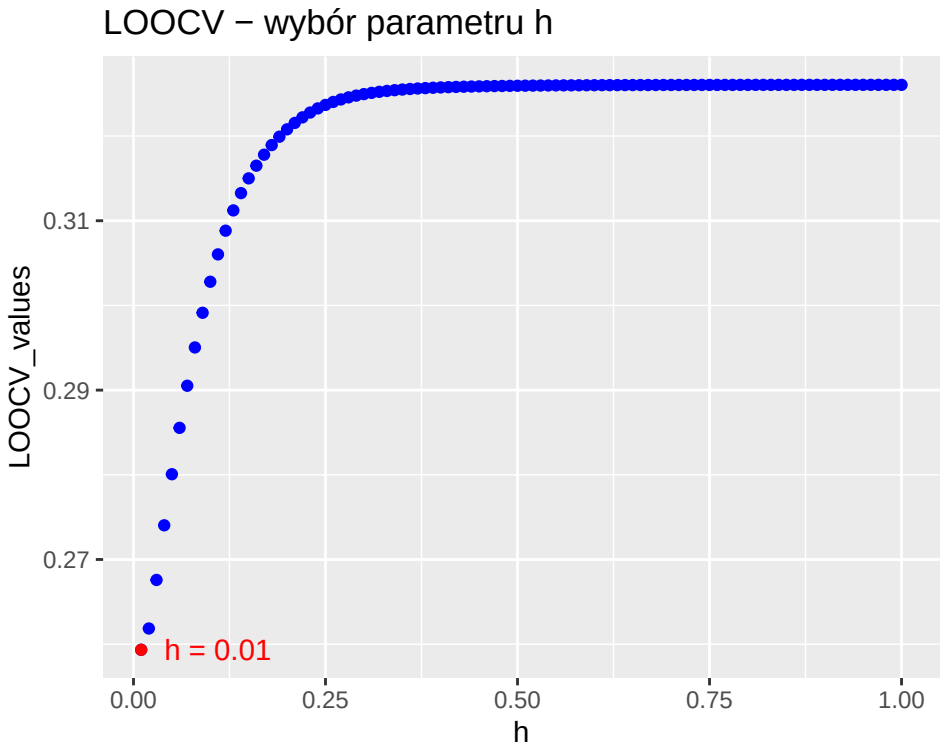
Rysunek 11: Wybór parametru h w oparciu o metodę LOOCV dla estymatora N-W dla sigma=0.5

Wykres nr 11 potwierdza optymalny wybór liczby $\hat{h}_n = 0.01$.

- Konstruujemy estymator funkcji regresji oparty na wielomianach lokalnych stopnia pierwszego oraz wybieramy h za pomocą LOOCV, wizualizując jednocześnie ten wybór.

```
LOOCV_fit_llp_0.5 <- np.cv(data=matrix(c(y_2, x_1), ncol=2, nrow=n_1),
                    h.seq=h,
                    w=c(0,1),
                    estimator='LLP',
                    kernel='gaussian'
)
h_opt_3 <- LOOCV_fit_llp_0.5$h.opt[2,1]
h_opt_3

## [1] 0.01

df_4 <- data.frame(h, LOOCV_fit_llp_0.5$CV)
colnames(df_4)[2] <- 'LOOCV_values'
ggplot(data=df_4, aes(x=h, y=LOOCV_values)) +
  geom_point(colour='blue') +
  geom_point(data=df_4[h == h_opt_3,],
             aes(x=h, y=LOOCV_values),
             colour = 'red'
  ) +
  geom_text(data=df_4[h == h_opt_3,],
```

```
            label="h = 0.01",
            aes(x=h+0.1, y=LOOCV_values),
            colour='red'
  ) +
  ggtitle('LOOCV - wybór parametru h')
```

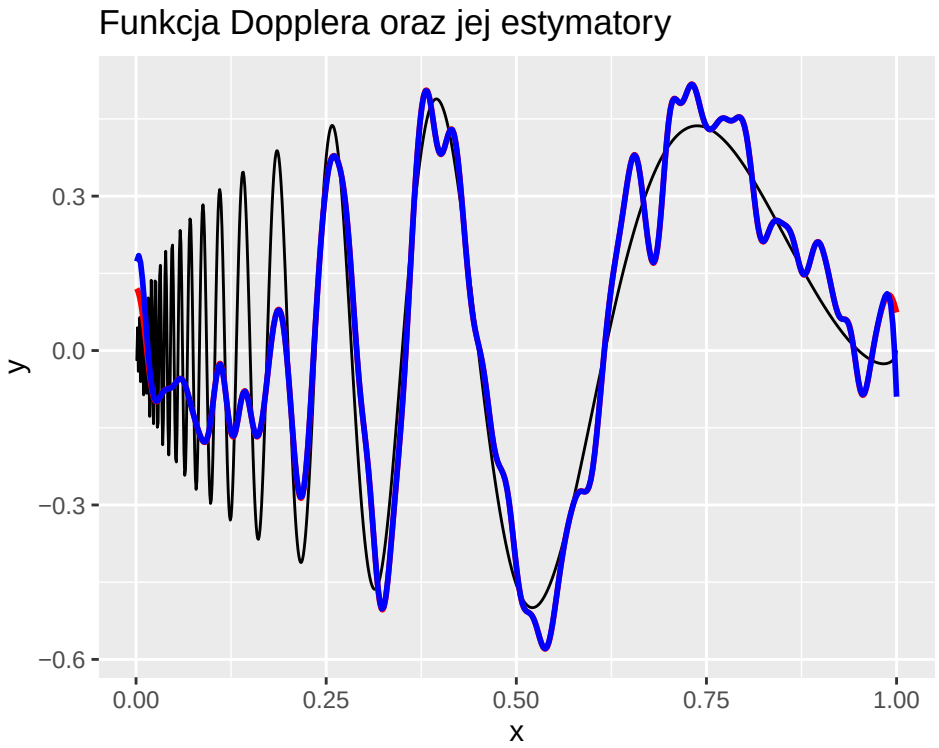


Rysunek 12: Wybór parametru h w oparciu o metodę LOOCV dla estymatora lokalnych wielomianów stopnia pierwszego dla sigma=0.5

Wykres nr 12 potwierdza optymalny wybór liczby $\hat{h}_n = 0.01$.

- Rysujemy wykres funkcji Dopplera wraz z jej estymatorami N-W oraz wielomianów lokalnych pierwszego stopnia.

```
df_5 <- data.frame(x_1, r_1, nw_estimator(x_1, y_2, h_opt_2))
df_6 <- data.frame(loc_pol(x_1, y_2, h_opt_3)$x,
                   loc_pol(x_1, y_2, h_opt_3)$y
)
colnames(df_5)[3] <- c('NW_estimator')
colnames(df_6)[1] <- 'x'
colnames(df_6)[2] <- 'local_polynomials_estimator'
ggplot(data=df_5, aes(x=x_1, y=r_1)) +
  geom_line() +
  geom_line(aes(x= x_1, y=NW_estimator), colour='red', lwd=1) +
  geom_line(data=df_6, aes(x=x, y=local_polynomials_estimator), colour='blue',
  ggtitle('Funkcja Dopplera oraz jej estymatory') +
  labs(x='x', y='y')
```



Rysunek 13: Funkcja Dopplera (kolor czarny) oraz jej estymatory N-W oraz wielomianów lokalnych stopnia pierwszego (odpowiednio kolor czerwony i niebieski) dla sigma=0.5

Dla mniejszej wartości parametru σ $(= 0.5)$ estymatory lepiej przybliżają badaną funkcję regresji. Dają one bardzo przybliżone efekty (nakładają się). Mimo wszystko nieco lepiej radzi sobie estymator wielomianów lokalnych, ponieważ lepiej przybliża "brzeg" funkcji Dopplera.

(b) $\sigma = 0.1$

- Generujemy obserwacje $Y_i, \quad i = 1, \ldots, n$ z nowym parametrem σ.

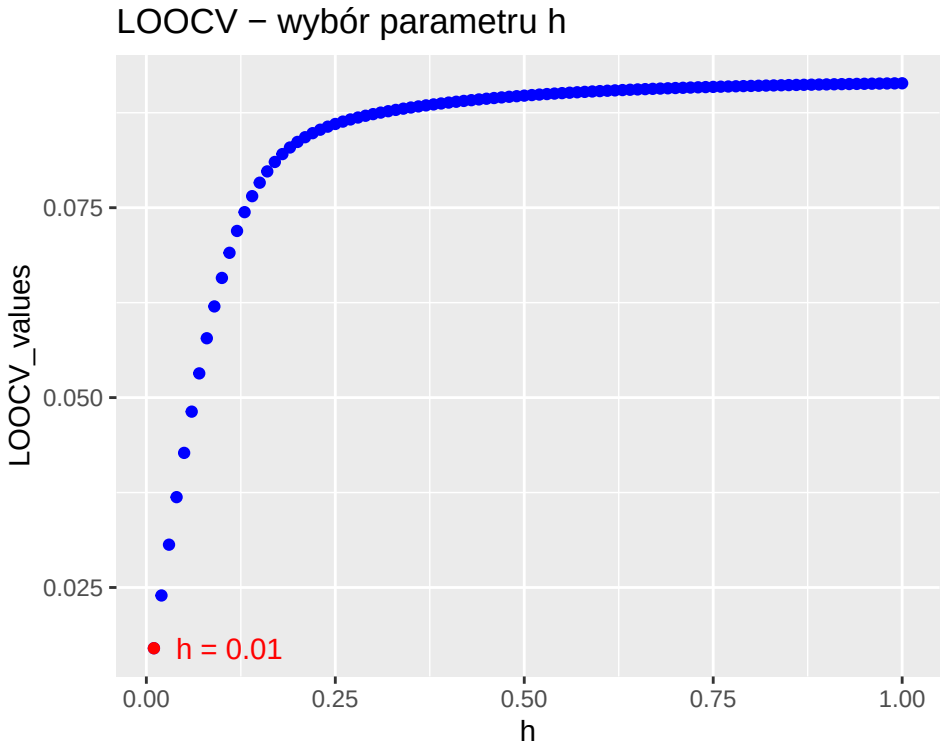
```
sigma_3 <- 0.1
y_3 <- r_1+rnorm(n_1, 0, sigma_3)
```

- Konstruujemy estymator Nadaraya-Watsona funkcji regresji oparty na jądrze gaussowskim, wyznaczając h przy pomocy LOOCV (znów użyjemy dla prostoty funkcji *np.cv*, zmieniając ustawienie *estimator='NW'*).

```
LOOCV_fit_nw_0.1 <- np.cv(data=matrix(c(y_3,x_1), ncol=2, nrow=n_1),
      h.seq=h,
      w=c(0,1),
      estimator='NW',
      kernel='gaussian'
)
h_opt_4 <- LOOCV_fit_nw_0.1$h.opt[2,1]
h_opt_4

## [1] 0.01
```

Sporządzamy także wykres estymatora $\hat{R}_{CV}(h)$.

```
df_6 <- data.frame(h, LOOCV_fit_nw_0.1$CV)
colnames(df_6)[2] <- 'LOOCV_values'
ggplot(data=df_6, aes(x=h, y=LOOCV_values)) +
  geom_point(colour='blue') +
  geom_point(data=df_6[h == h_opt_4,],
             aes(x=h, y=LOOCV_values),
             colour = 'red'
  ) +
  geom_text(data=df_6[h == h_opt_4,],
            label="h = 0.01",
            aes(x=h+0.1, y=LOOCV_values),
            colour='red'
  ) +
  ggtitle('LOOCV - wybór parametru h')
```



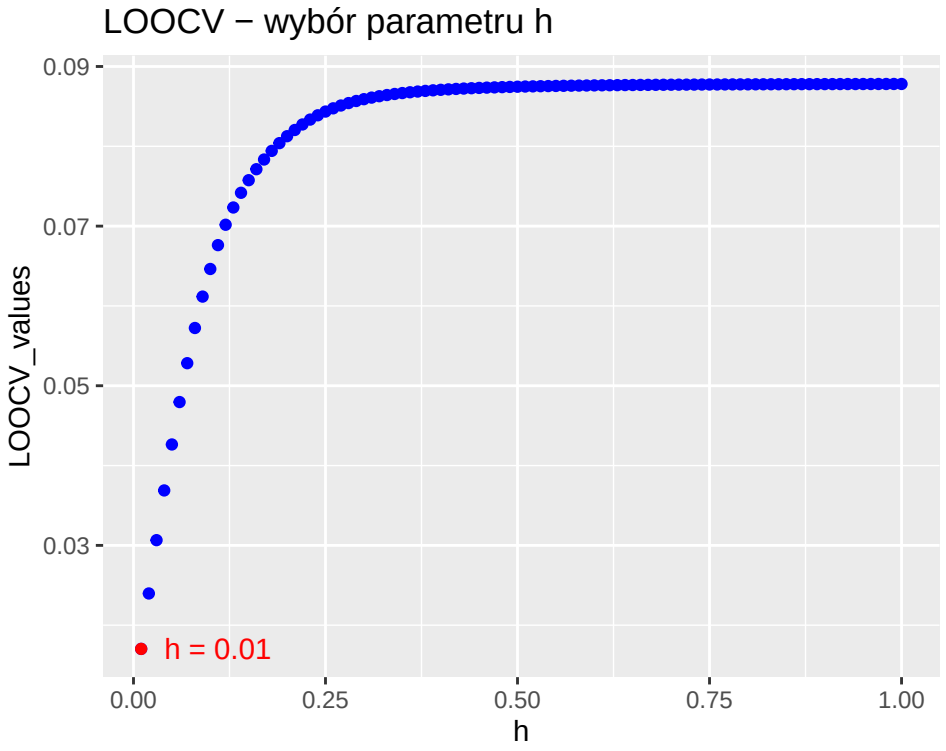
Rysunek 14: Wybór parametru h w oparciu o metodę LOOCV dla estymatora N-W dla sigma=0.1

Wykres nr 14 potwierdza optymalny wybór liczby $\hat{h}_n = 0.01$.

- Konstruujemy estymator funkcji regresji oparty na wielomianach lokalnych stopnia pierwszego oraz wybieramy h za pomocą LOOCV, wizualizując jednocześnie ten wybór.

```
LOOCV_fit_llp_0.1 <- np.cv(data=matrix(c(y_3, x_1), ncol=2, nrow=n_1),
                           h.seq=h,
```

```
                              w=c(0,1),
                              estimator='LLP',
                              kernel='gaussian'
)

h_opt_5 <- LOOCV_fit_llp_0.1$h.opt[2,1]
h_opt_5

## [1] 0.01

df_7 <- data.frame(h, LOOCV_fit_llp_0.1$CV)
colnames(df_7)[2] <- 'LOOCV_values'
ggplot(data=df_7, aes(x=h, y=LOOCV_values)) +
  geom_point(colour='blue') +
  geom_point(data=df_7[h == h_opt_5,],
             aes(x=h, y=LOOCV_values),
             colour = 'red'
  ) +
  geom_text(data=df_7[h == h_opt_5,],
            label="h = 0.01",
            aes(x=h+0.1, y=LOOCV_values),
            colour='red'
  ) +
  ggtitle('LOOCV - wybór parametru h')
```

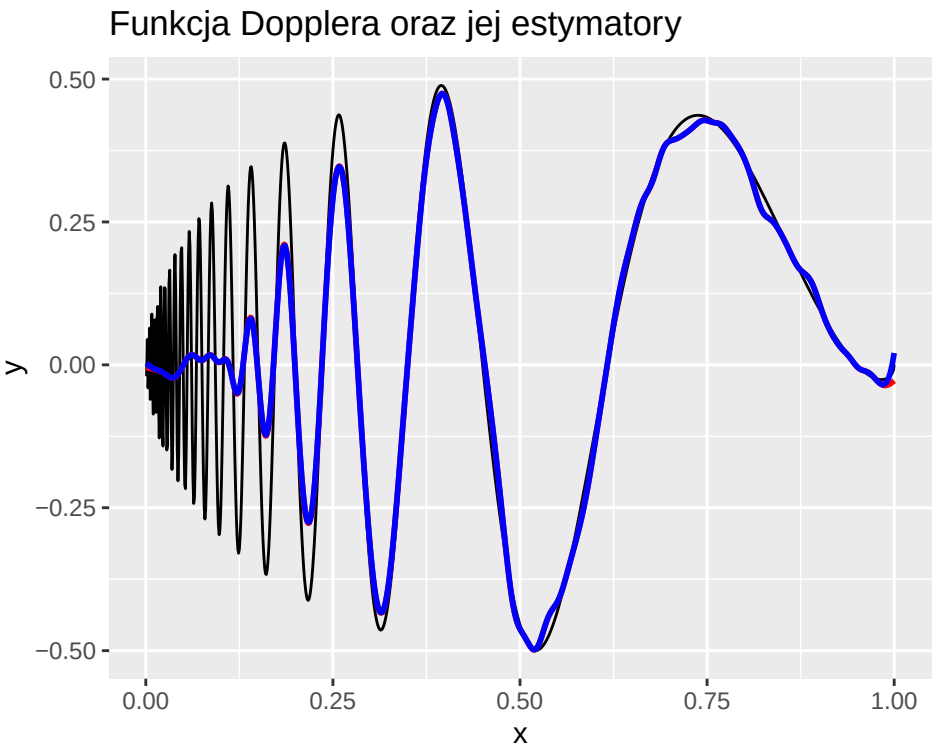


Rysunek 15: Wybór parametru h w oparciu o metodę LOOCV dla estymatora lokalnych wielomianów stopnia pierwszego dla sigma=0.1

Wykres nr 15 potwierdza optymalny wybór liczby $\hat{h}_n = 0.01$.

- Rysujemy wykres funkcji Dopplera wraz z jej estymatorami N-W oraz wielomianów lokalnych pierwszego stopnia.

```
df_8 <- data.frame(x_1, r_1, nw_estimator(x_1, y_3, h_opt_4))
df_9 <- data.frame(loc_pol(x_1, y_3, h_opt_5)$x,
                   loc_pol(x_1, y_3, h_opt_5)$y
)
colnames(df_8)[3] <- c('NW_estimator')
colnames(df_9)[1] <- 'x'
colnames(df_9)[2] <- 'local_polynomials_estimator'
ggplot(data=df_8, aes(x=x_1, y=r_1)) +
  geom_line() +
  geom_line(aes(x= x_1, y=NW_estimator), colour='red', lwd=1) +
  geom_line(data=df_9, aes(x=x, y=local_polynomials_estimator), colour='blue',
  ggtitle('Funkcja Dopplera oraz jej estymatory') +
  labs(x='x', y='y')
```



Rysunek 16: Funkcja Dopplera (kolor czarny) oraz jej estymatory N-W oraz wielomianów lokalnych stopnia pierwszego (odpowiednio kolor czerwony i niebieski) dla sigma=0.1

Wcześniejsze obserwacje pozostają w mocy - obserwujemy lepsze dopasowanie estymatorów do funkcji regresji oraz estymatory te działają z podobnym efektem, nawet na brzegach krzywej.

Wyciągnijmy wnioski z przeprowadzonej analizy z zadania nr 5.

Wniosek 4 *Im większe zakłócenie, tym słabsza estymacja (o tym, jak bardzo słaba decyduje także kształt estymowanej krzywej).*

Wniosek 5 *Estymacja nieparametryczna potrafi zawodzić w przypadku estymacji nieregularnych funkcji.*

Wniosek 6 *W przypadku wyboru optymalnego parametru h dla estymatorów Nadaraya-Watsona oraz lokalnych wielomianów stopnia pierwszego, metoda* leave-one-out cross-validation *dąży do wybrania najmniejszego h spośród rozważanej "siatki punktów" (a przynajmniej wtedy, gdy w estymatorach tych rozważamy jądro gaussowskie - tylko ten przypadek zbadaliśmy).*