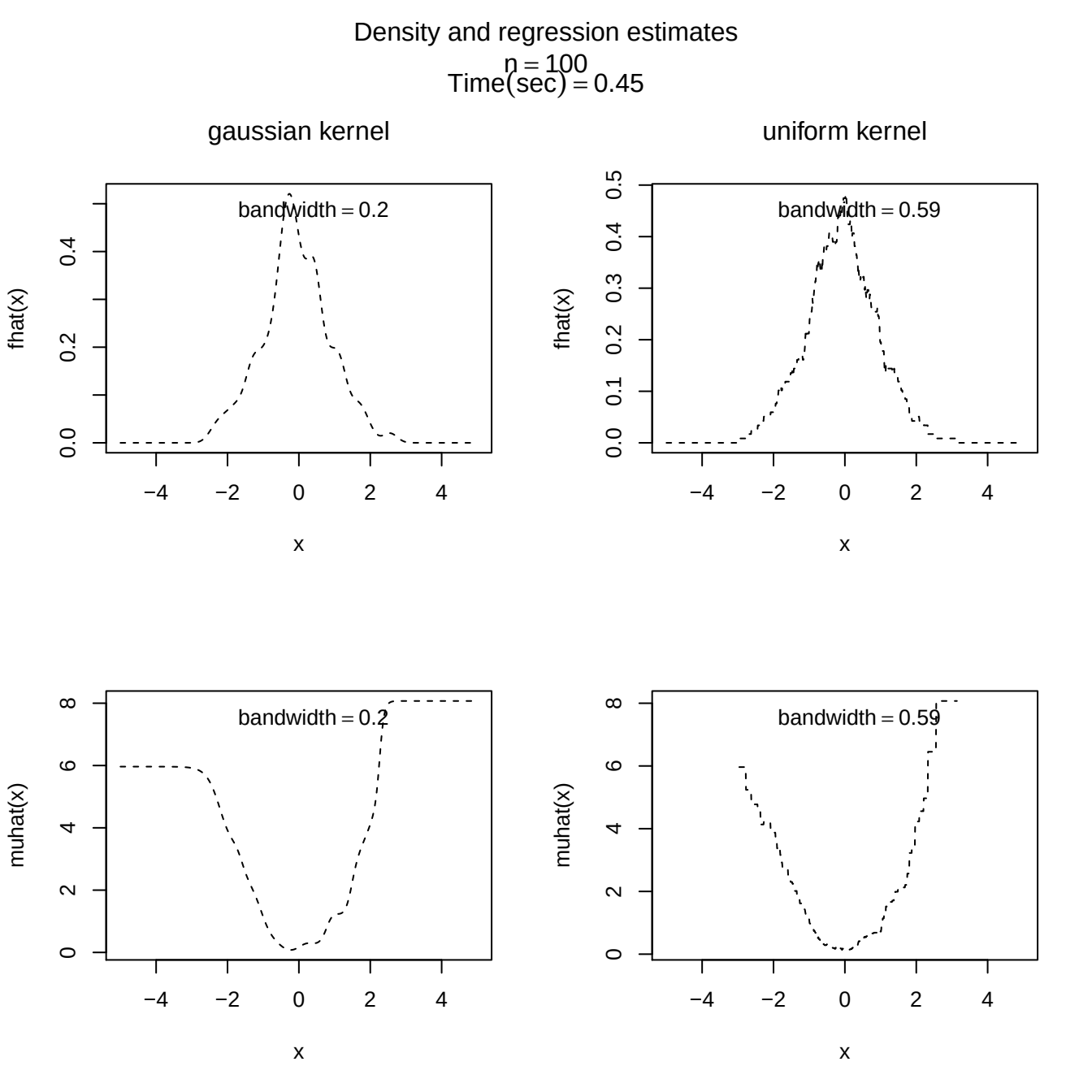
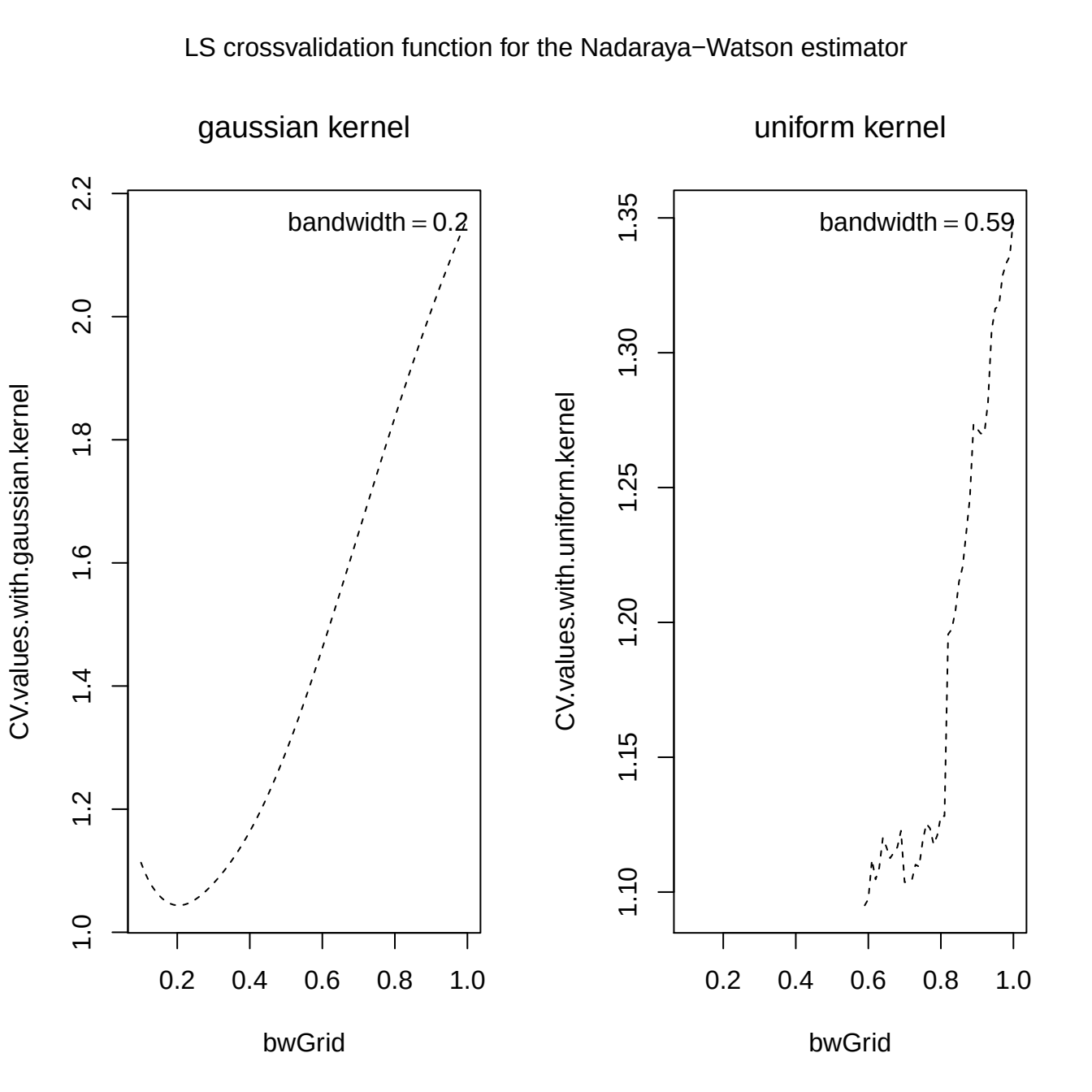
```r
# R code for kernel estimator of pdf_X and E(Y|X).
# Gautam Tripathi
# March 03, 2017

rm(list = ls()) # Clear workspace.
ptm = proc.time() # Start clock.
set.seed(12345678) # Set seed for replication.

n = 100 # Number of observations.

# Generate the data.
X = rnorm(n, mean=0, sd=1)
U = rnorm(n, mean=0, sd=1)
Y = X^2 + U

# Generate the x-values at which pdf_X(x) and E(Y|X=x) are to be estimated
Xgrid = seq(-5,5,by=0.01) # Array of length m := length(Xgrid).

gaussian.kernel = function(u)
  # Gaussian kernel.
{
  return(dnorm(u))
}

uniform.kernel = function(u)
  # Uniform kernel.
{
  return(0.5*ifelse((u >= -1)&(u <= 1), 1, 0))
}

my.ksmooth = function(Ydata,Xdata,xvalues,N,h,kernel)
  # Function for estimating pdf_X(x) and E(Y|X=x).
{
  Dmatrix = outer(Xdata, xvalues, "-") # n x m
  Kmatrix = kernel(Dmatrix/h)
  den = colSums(Kmatrix)/(N*h)
  num = colSums(sweep(Kmatrix, 1, Ydata, "*"))/(N*h) # Sweeps array Ydata across
      columns of Kmatrix.
  est = list(fhat=den, muhat=num/den)
  return(est)
}

muhat.LOO = function(Ydata,Xdata,N,h,kernel)
  # Function for estimating the LOO estimator of muhat.
  # Returns muhat_{-1}(X_1), ..., muhat_{-n}(X_n).
{
  Dmatrix = outer(Xdata, Xdata, "-") # n x n
  Kmatrix = kernel(Dmatrix/h)
```

```
48    diag(Kmatrix) = 0 # Set diagonal elements of Kmatrix to 0.
49    num = colSums(sweep(Kmatrix, 1, Ydata, "*"))/((N-1)*h)
50    den = colSums(Kmatrix)/((N-1)*h)
51    return(num/den)
52  }
53
54  cv.muhat = function(Ydata,Xdata,N,h,kernel)
55    # Cross-validation function for Nadaraya-Watson estimator of E(Y|X).
56  {
57    muhat.LOO.values = muhat.LOO(Ydata,Xdata,N,h,kernel)
58    ase.LOO = mean((Ydata - muhat.LOO.values)^2)
59    return(ase.LOO)
60  }
61
62  bwGrid = seq(0.1,1,by=0.01)
63  CV.values.with.gaussian.kernel = matrix(NA, length(bwGrid), 1)
64  CV.values.with.uniform.kernel = matrix(NA, length(bwGrid), 1)
65
66  for (i in 1:length(bwGrid))
67  {
68    CV.values.with.gaussian.kernel[i,1] = cv.muhat(Y,X,n,bwGrid[i],gaussian.kernel
        )
69    CV.values.with.uniform.kernel[i,1] = cv.muhat(Y,X,n,bwGrid[i],uniform.kernel)
70  }
71
72  # Crossvalidated bandwidths for muhat.
73  cv.bw.with.gaussian.kernel = bwGrid[which.min(CV.values.with.gaussian.kernel)]
74  cv.bw.with.uniform.kernel = bwGrid[which.min(CV.values.with.uniform.kernel)]
75
76  # Obtain fhat and muhat.
77  fhat.muhat.with.gaussian.kernel = my.ksmooth(Y,X,Xgrid,n,cv.bw.with.gaussian.
      kernel,gaussian.kernel)
78  fhat.with.gaussian.kernel = fhat.muhat.with.gaussian.kernel$fhat
79  muhat.with.gaussian.kernel = fhat.muhat.with.gaussian.kernel$muhat
80
81  fhat.muhat.with.uniform.kernel = my.ksmooth(Y,X,Xgrid,n,cv.bw.with.uniform.
      kernel,uniform.kernel)
82  fhat.with.uniform.kernel = fhat.muhat.with.uniform.kernel$fhat
83  muhat.with.uniform.kernel = fhat.muhat.with.uniform.kernel$muhat
84
85  runningtime = proc.time() - ptm # Record running time.
86  cat("total run time (sec) =", round(runningtime[3],1), "\n")
87
88  # Format data to be reported in the plots.
89  bw.gaussian = bquote(bandwidth == .(cv.bw.with.gaussian.kernel))
90  bw.uniform = bquote(bandwidth == .(cv.bw.with.uniform.kernel))
91  report.n = bquote(n == .(n))
92  report.runtime = bquote(Time(sec) == .(runningtime))
```

```
93
94 # Plot fhat and muhat to a pdf file.
95 pdf(file="den+reg-5+LOO-graphs.pdf") # Open PDF file for writing.
96
97 par(oma=c(0,0,3,0)) # Set margins: bottom=left=right=0, top=3 for main title.
98 par(mfrow=c(2,2)) # 2 x 2 plots.
99
100 # Begin plots.
101 plot(Xgrid, fhat.with.gaussian.kernel, type="l", lty="dashed", xlab="x", ylab="fhat
    (x)")
102 title(main="gaussian kernel", font.main=1) # Individual title.
103 legend('top', legend=bw.gaussian, bty='n')
104
105 plot(Xgrid, fhat.with.uniform.kernel, type="l", lty="dashed", xlab="x", ylab="fhat(x
    )")
106 title(main="uniform kernel", font.main=1)
107 legend('top', legend=bw.uniform, bty='n')
108
109 plot(Xgrid, muhat.with.gaussian.kernel, type="l", lty="dashed", xlab="x", ylab="
    muhat(x)")
110 legend('top', legend=bw.gaussian, bty='n')
111
112 plot(Xgrid, muhat.with.uniform.kernel, type="l", lty="dashed", xlab="x", ylab="muhat
    (x)")
113 legend('top', legend=bw.uniform, bty='n')
114
115 top.plot.title = list("Density and regression estimates", report.n, report.
    runtime)
116 mtext(do.call(expression, top.plot.title), outer=TRUE, line=1:-1) # Main title on
    top.
117 dev.off() # Close file.
118
119 # Plot cross-validation functions to a pdf file.
120 pdf(file="den+reg-5+LOO-cv.pdf")
121
122 par(oma=c(0,0,2,0)) # Set margins: bottom=left=right=0, top=2 for main title.
123 par(mfrow=c(1,2)) # 1 x 2 plots.
124
125 plot(bwGrid, CV.values.with.gaussian.kernel, type="l", lty="dashed")
126 title(main="gaussian kernel", font.main=1)
127 legend('topright', legend=bw.gaussian, bty='n')
128
129 plot(bwGrid, CV.values.with.uniform.kernel, type="l", lty="dashed")
130 title(main="uniform kernel", font.main=1)
131 legend('topright', legend=bw.uniform, bty='n')
132 mtext("LS crossvalidation function for the Nadaraya-Watson estimator", outer=
    TRUE)
133 dev.off()
```


Density and regression estimates
n = 100
Time(sec) = 0.45
gaussian kernel
uniform kernel
bandwidth = 0.2
bandwidth = 0.59
fhat(x)
muhat(x)
x
0.0
0.1
0.2
0.3
0.4
0.5
0
2
4
6
8
−4
−2
0
2
4


LS crossvalidation function for the Nadaraya−Watson estimator
gaussian kernel
uniform kernel
bandwidth = 0.2
bandwidth = 0.59
CV.values.with.gaussian.kernel
CV.values.with.uniform.kernel
bwGrid
bwGrid
1.0
1.2
1.4
1.6
1.8
2.0
2.2
1.10
1.15
1.20
1.25
1.30
1.35
0.2
0.4
0.6
0.8
1.0