Assignment II

2023-10-28

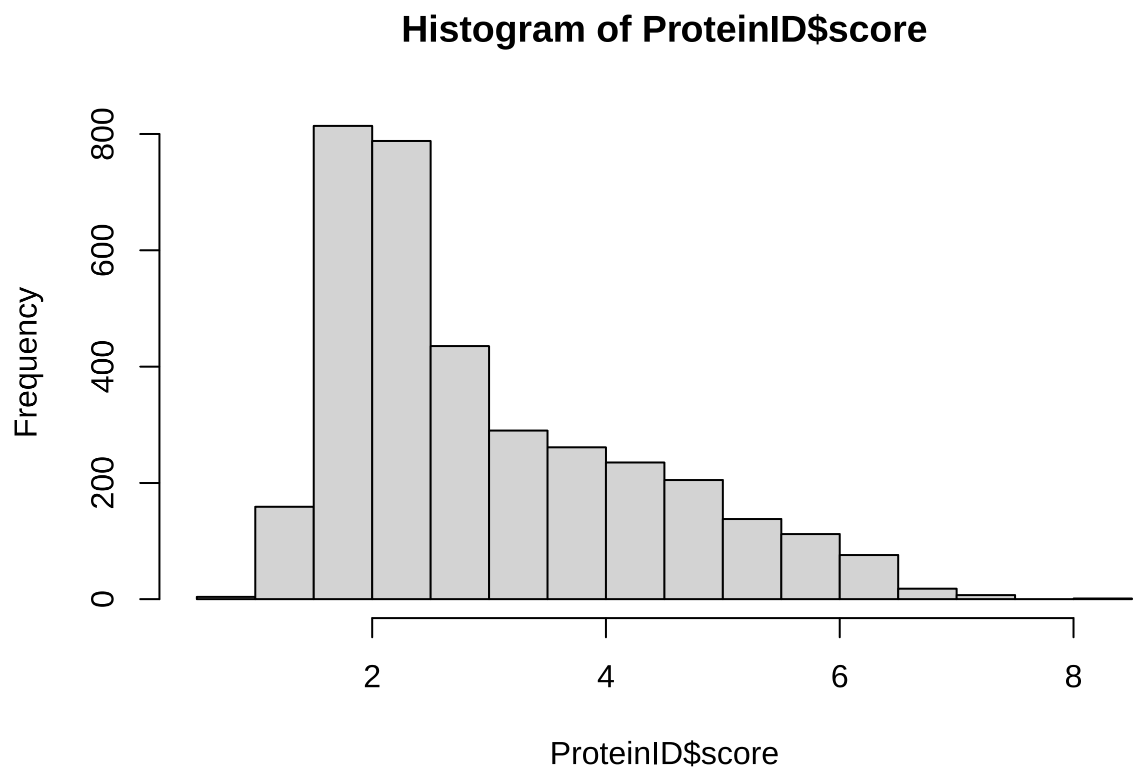
R Markdown

1. Make a histogram (R: hist) with a bin width of 1 of the protein score data (second column). (5%)

```
# Import data
ProteinID <- read.csv("C:/deskt/DIR/7th/Desktop/Semester 1/Foundations in Math & Stats/Week II/ProteinI
# Check head
head(ProteinID)
```

```
##   index score                                    peptide
## 1     1 4.344                          R.AMYTQAVQNIR.V
## 2     2 2.024                         K.DSDVEVYNIIKK.E
## 3     3 2.621 K.LGQIPEGEGASEQGM[147.04]ARHSHSGLR.A
## 4     4 2.385           K.FFQLFKPIM[147.04]INLPR.D
## 5     5 1.711                 K.KILHIGELVNTIDYIR.R
## 6     6 2.730                            K.KYLYEIAR.R
##
## 1
## 2
## 3
## 4 RRRRRIPI:IPI00024364.1|SWISS-PROT:Q92973|REFSEQ_NP:NP_694858;NP_002261|ENSEMBL:ENSP00000336712;ENS
## 5
## 6
```

```
hist(ProteinID$score)
```



2. Generate a sequence (R: seq) of values from 0 to 10 with a stepwidth of 0.01. Generate the density of the original mixture of normal distributions (R: dnorm) at all positions of this sequence.(5%)

```
# Generate a sequence of values from 0 to 10 with a stepwidth of 0.01
x <- seq(0, 10, by = 0.01)

# Generate the density of the original mixture of normal distributions
original_density <- dnorm(x, mean = 2, sd = 0.4) + dnorm(x, mean = 4, sd = 1.2)
```

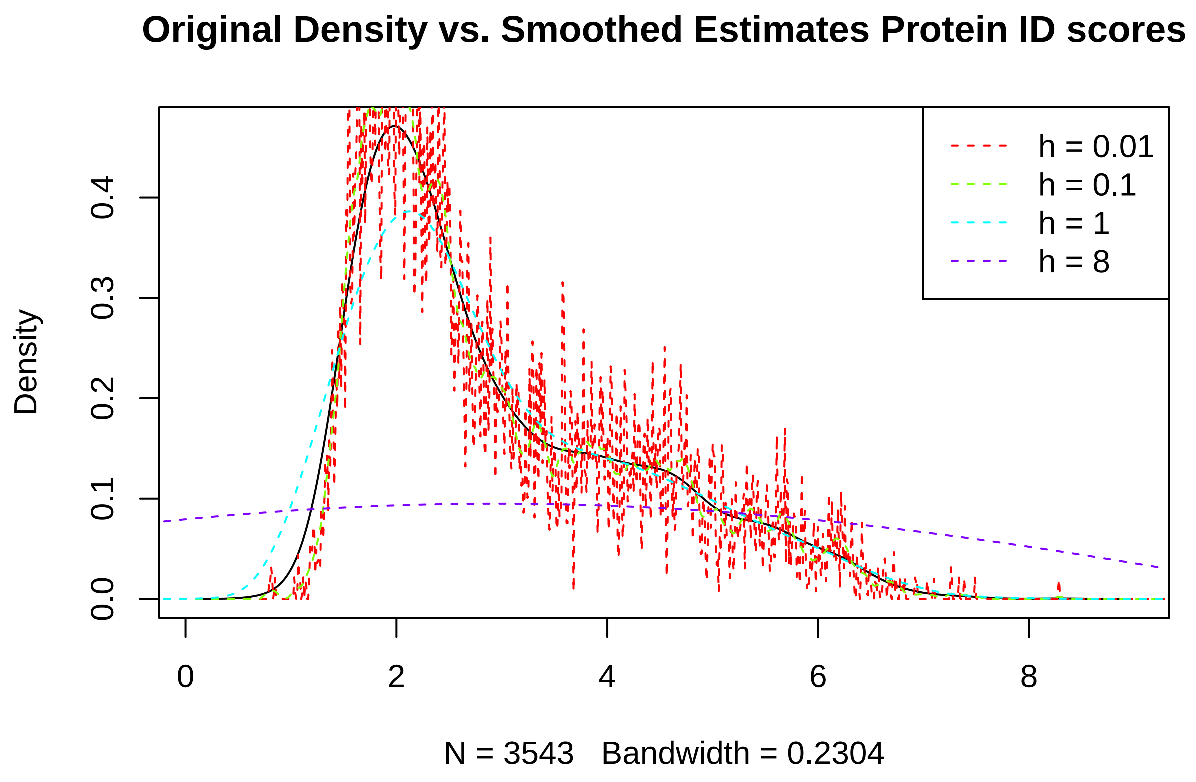
3. Write a function which smooths the experimental data for all positions of the sequence with the given kernel and bandwidths. I.e., for each bandwidth evaluate the kernel density estimator at each sequence value. (20%)

```
# Define the custom kernel function
kernel <- function(x) {
  (pi/4) * cos((pi/2) * x) * (abs(x) <= 1)
}

# Create a function to perform smoothing
smooth_data <- function(data, bandwidths) {
  smoothed_densities <- list()
```

```r
  for (h in bandwidths) {
    # Calculate the smoothed density estimate using the custom kernel
    x <- seq(min(data) - 1, max(data) + 1, length.out = 1000)
    kde_estimate <- sapply(x, function(xi) mean(kernel((data - xi) / h)) / h)

    smoothed_densities[[as.character(h)]] <- list(x = x, y = kde_estimate)
  }

  return(smoothed_densities)
}

# Define bandwidths
bandwidths <- c(0.01, 0.1, 1, 8)
experimental_data <- c(ProteinID$score)

# Smooth the experimental data for different bandwidths
smoothed_densities <- smooth_data(experimental_data, bandwidths)
```

4. Make (a) suitable plot(s) to compare the original density with the various smoothed estimates (recommended commands: R: plot, lines, legend).

```r
# Plot the original density and smoothed estimates
plot(density(experimental_data), main = "Original Density vs. Smoothed Estimates Protein ID scores")

# Add smoothed estimates to the plot
legend_labels <- character(0)
for (h in bandwidths) {
  lines(smoothed_densities[[as.character(h)]]$x, smoothed_densities[[as.character(h)]]$y, col = rainbow
  legend_labels <- c(legend_labels, paste("h =", h))
}

# Add a legend
legend("topright", legend = legend_labels, col = rainbow(length(bandwidths)), lty = 2)
```



### 5. Compute the mean squared error (over all positions of the sequence) for each of the 4 bandwidths. What is the best bandwidth in this example?

```r
# Create a function to perform smoothing and calculate MSE
smooth_data_with_mse <- function(data, bandwidths) {
  smoothed_densities <- list()
  mse_values <- numeric(length(bandwidths))

  for (i in 1:length(bandwidths)) {
    h <- bandwidths[i]
    # Calculate the smoothed density estimate using the custom kernel
    x <- seq(min(data) - 1, max(data) + 1, length.out = 1000)
    kde_estimate <- sapply(x, function(xi) mean(kernel((data - xi) / h)) / h)

    smoothed_densities[[i]] <- list(x = x, y = kde_estimate)

    # Calculate the MSE
    mse_values[i] <- mean((kde_estimate - dnorm(x))^2)  # Compare with a normal distribution

    cat("MSE for bandwidth =", h, ":", mse_values[i], "\n")
  }

  return(list(smoothed_densities = smoothed_densities, mse_values = mse_values))
}

# Define bandwidths
bandwidths <- c(0.01, 0.1, 1, 8)
```

```r
experimental_data <- c(ProteinID$score)

# Smooth the experimental data for different bandwidths and calculate MSE
result <- smooth_data_with_mse(experimental_data, bandwidths)
```

```
## MSE for bandwidth = 0.01 : 0.04325274
## MSE for bandwidth = 0.1 : 0.04133246
## MSE for bandwidth = 1 : 0.033866
## MSE for bandwidth = 8 : 0.01427445
```

```r
# Access the smoothed densities and MSE values
smoothed_densities <- result$smoothed_densities
mse_values <- result$mse_values
```

The bandwidth of 8 has the smallest MSE (0.01427445), which means it provides the best fit to the data according to the MSE criterion.

6. Try more values for the bandwidth and find out what the optimal bandwidth is (i.e. minimizing the MSE).

```r
# Define a sequence of bandwidth values to try
bandwidths_to_try <- seq(0.01, 10, by = 0.01)

# Function to calculate MSE for a given bandwidth
calculate_mse <- function(data, bandwidth) {
  x <- seq(min(data) - 1, max(data) + 1, length.out = 50)
  kde_estimate <- sapply(x, function(xi) mean(kernel((data - xi) / bandwidth)) / bandwidth)
  mse <- mean((kde_estimate - dnorm(x))^2)
  return(mse)
}

# Initialize variables to store the best bandwidth and its corresponding MSE
best_bandwidth <- NULL
min_mse <- Inf

# Iterate over bandwidth values and calculate MSE
for (bw in bandwidths_to_try) {
  mse <- calculate_mse(experimental_data, bw)

  # Update best bandwidth if the current MSE is smaller
  if (mse < min_mse) {
    min_mse <- mse
    best_bandwidth <- bw
  }
}

cat("Optimal Bandwidth:", best_bandwidth, "\n")
```

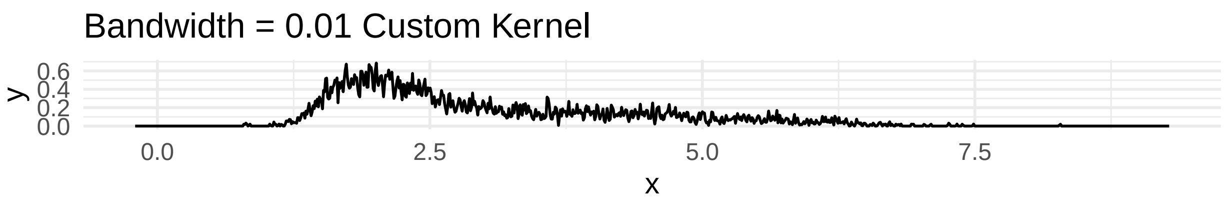
```
## Optimal Bandwidth: 8.92
```

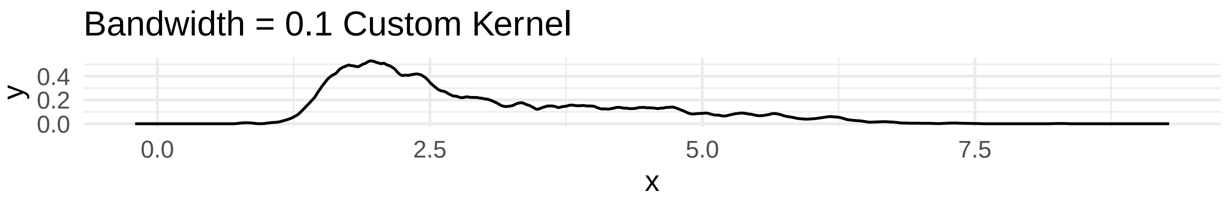
```r
cat("Minimum MSE:", min_mse, "\n")
```

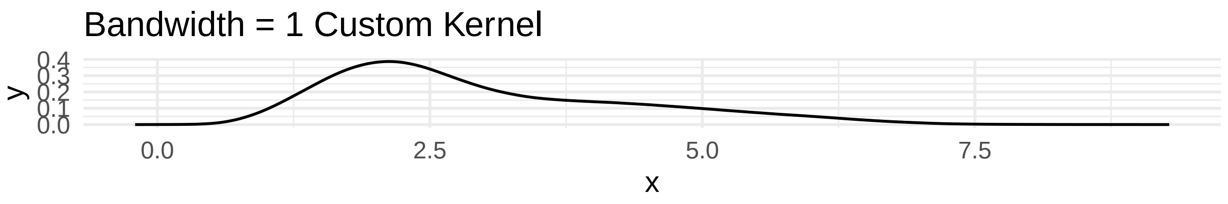
```
## Minimum MSE: 0.01490455
```

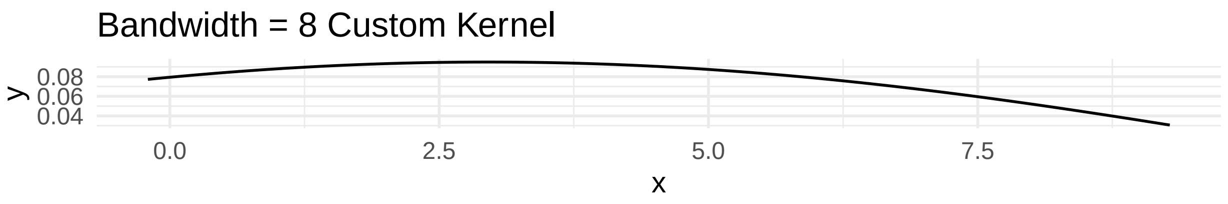
7. Compare your results with KDE libraries in R (density) or Python (sklearn.neighbors.KernelDensity), and make some plots to see the effect of different bandwidths and kernel functions. Does the choice of kernel functions K or the bandwidths h affect density estimation more?

```r
library(ggplot2)
library(gridExtra)

# Create a function to plot density estimates for different bandwidths using the custom kernel
plot_custom_kernel_density <- function(data, bandwidths) {
  plots <- list()

  for (h in bandwidths) {
    x <- seq(min(data) - 1, max(data) + 1, length.out = 1000)
    kde_estimate <- sapply(x, function(xi) mean(kernel((data - xi) / h)) / h)

    plot_data <- data.frame(x = x, y = kde_estimate)
    title <- paste("Bandwidth =", h, "Custom Kernel")

    p <- ggplot(plot_data, aes(x, y)) +
      geom_line() +
      labs(title = title) +
      theme_minimal()

    plots[[length(plots) + 1]] <- p
  }

  return(plots)
}

# Define bandwidths to test
bandwidths <- c(0.01, 0.1, 1, 8)

# Create density plots using your custom kernel
custom_kernel_density_plots <- plot_custom_kernel_density(experimental_data, bandwidths)

# Arrange the plots using grid.arrange from gridExtra
grid.arrange(grobs = custom_kernel_density_plots, ncol = 1)
```









Yes the choices have an effect on the results and are an important part of the research. It's important to experiment with different combinations of kernel functions and bandwidths to find the best fit for our specific data and analysis objectives.