SHC 798 Assignment 1, 2025

Richard Lubega

2025-07-14

SHC 798 Assignment 1, 2025

Part 2: Data Smoothing

Traffic Flow Data Analysis

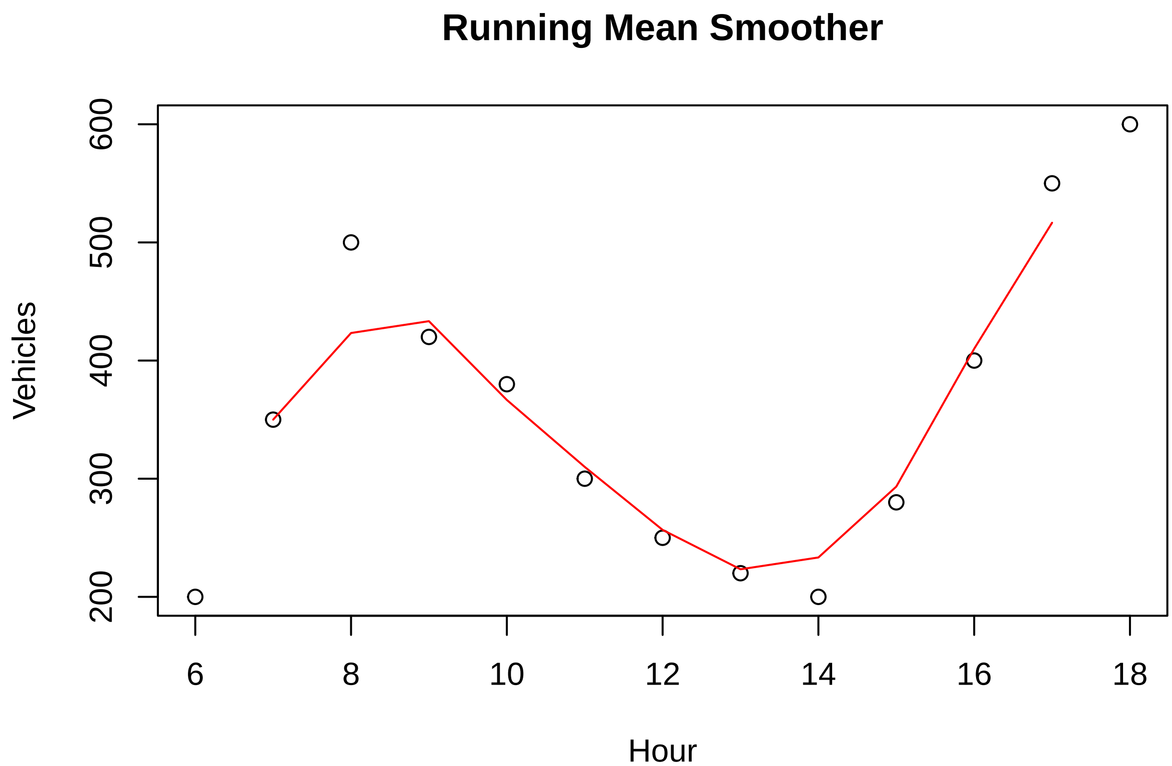
```r
# Traffic flow data
hour <- 6:18
vehicles <- c(200, 350, 500, 420, 380, 300, 250, 220, 200, 280, 400, 550, 600)
traffic <- data.frame(hour, vehicles)
```

(a) Validating in R

```r
# Compute running mean using window width of 3
traffic$smoothed <- stats::filter(traffic$vehicles, rep(1/3, 3), sides = 2)
print(traffic)
```

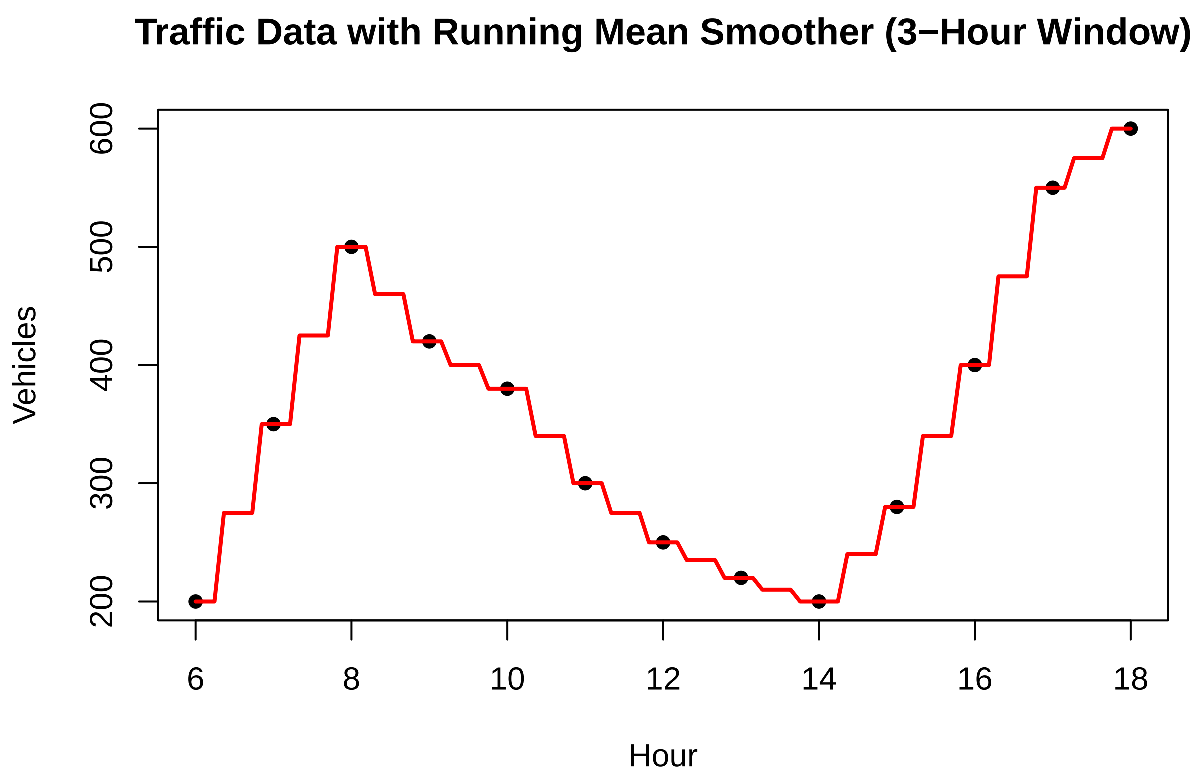
```
##    hour vehicles smoothed
## 1     6      200       NA
## 2     7      350 350.0000
## 3     8      500 423.3333
## 4     9      420 433.3333
## 5    10      380 366.6667
## 6    11      300 310.0000
## 7    12      250 256.6667
## 8    13      220 223.3333
## 9    14      200 233.3333
## 10   15      280 293.3333
## 11   16      400 410.0000
## 12   17      550 516.6667
## 13   18      600       NA
```

```r
# Scatter Plot
plot(hour, vehicles, type = "p", main = "Running Mean Smoother", xlab = "Hour", ylab = "Vehicles")
lines(traffic$hour, traffic$smoothed, type = "l", col = "red")
```



(b) Using R to compute a running mean smoother using ksmooth()

```r
# Compute running mean smoother using ksmooth
smoothed <- ksmooth(traffic$hour, traffic$vehicles, kernel = "box", bandwidth = 1.5)

# Create a data frame with smoothed values
traffic_smoothed <- data.frame(hour = smoothed$x, vehicles_smoothed = smoothed$y)

# Plot scatter plot of original data and overlay the smoothed line
plot(traffic$hour, traffic$vehicles,
     xlab = "Hour", ylab = "Vehicles",
     main = "Traffic Data with Running Mean Smoother (3-Hour Window)",
     pch = 16, col = "black")  # Scatter plot
lines(traffic_smoothed$hour, traffic_smoothed$vehicles_smoothed,
      col = "red", lwd = 2)  # Smoothed line
```



(c) Validating in R

```
# Checking the Gaussian kernel smoother (does not have the normalization constant)
gaussian_kernel <- function(xi, x, y, h) {
  weights <- exp(-((x - xi)^2) / (2 * h^2))
  sum(weights * y) / sum(weights)}

# Compute values
xi_values <- 6:18
kernel_values <- sapply(xi_values, function(xi) gaussian_kernel(xi, hour, vehicles, h = 2))

# # Validating with h = 2
validation <- data.frame(xi = xi_values,   manual_values = round(kernel_values, 4))
print(validation)
```

```
##    xi manual_values
## 1   6      338.0665
## 2   7      360.0905
## 3   8      372.6262
## 4   9      369.2072
## 5  10      348.6222
## 6  11      318.1419
## 7  12      291.1620
## 8  13      281.2593
## 9  14      296.4157
```

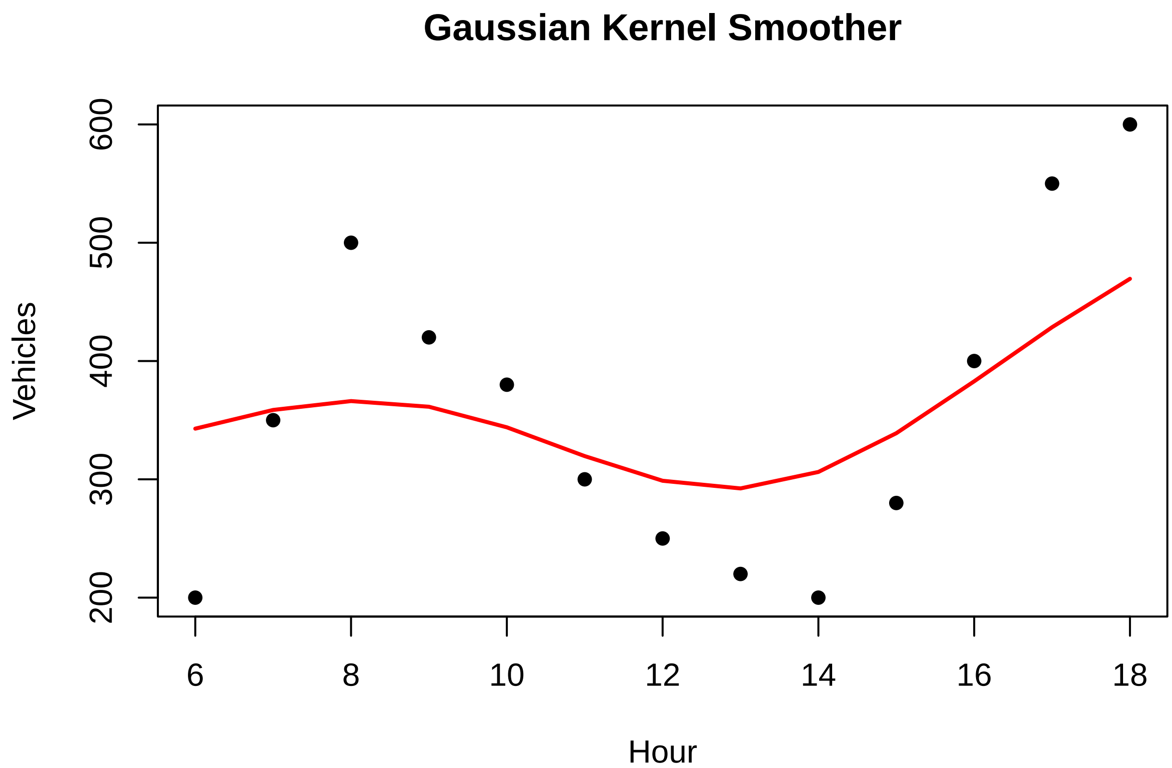
```
## 10 15      335.1786
## 11 16      387.3647
## 12 17      440.2893
## 13 18      485.3281
```

(d) Create the Gaussian kernel smoother with R, using the function ksmooth()

```r
# Gaussian kernel smoother with bandwidth = 2
smoothed <- ksmooth(traffic$hour, traffic$vehicles, kernel = "normal", bandwidth = 6, x.points = traffic
traffic_smoothed <- data.frame(hour = smoothed$x, vehicles_smoothed = smoothed$y)
print(traffic_smoothed)
```

```
##    hour vehicles_smoothed
## 1     6          342.8414
## 2     7          358.6636
## 3     8          366.1537
## 4     9          361.3546
## 5    10          343.9766
## 6    11          319.6306
## 7    12          298.7996
## 8    13          292.3215
## 9    14          306.2371
## 10   15          338.9925
## 11   16          382.8491
## 12   17          428.5728
## 13   18          469.5297
```

```r
# Plot original data and smoothed curve
plot(traffic$hour, traffic$vehicles, xlab = "Hour", ylab = "Vehicles", main = "Gaussian Kernel Smoother
     pch = 16, col = "black")
lines(traffic_smoothed$hour, traffic_smoothed$vehicles_smoothed, col = "red", lwd = 2)
```



(e) **Using LOESS smoother**

```
# Defining LOESS smoothers with varying degrees and spans
loess_1_03 <- loess(vehicles ~ hour, data = traffic, degree = 1, span = 0.3)
```

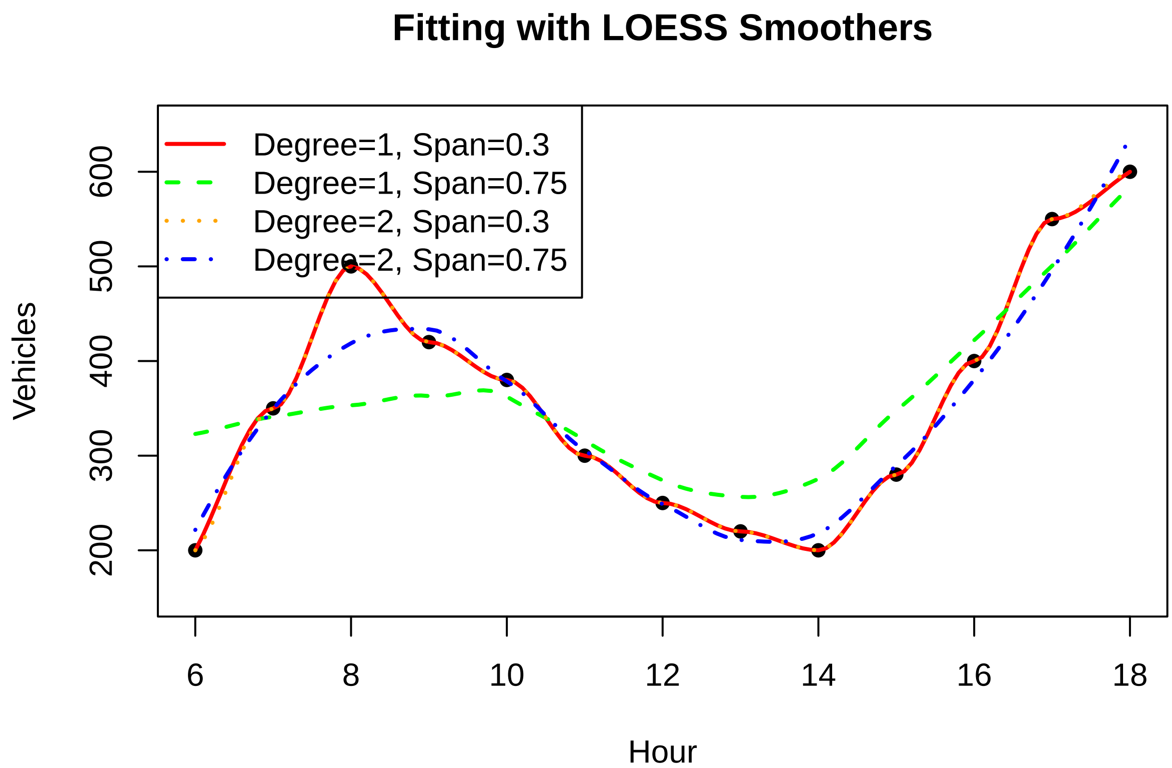
```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric,
## : pseudoinverse used at 12

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric,
## : neighborhood radius 1

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric,
## : reciprocal condition number -0

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric,
## : There are other near singularities as well. 1
```

```
loess_1_075 <- loess(vehicles ~ hour, data = traffic, degree = 1, span = 0.75)
loess_2_03 <- loess(vehicles ~ hour, data = traffic, degree = 2, span = 0.3)
```

```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric,
## : span too small.  fewer data values than degrees of freedom.
```

```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric,
## : pseudoinverse used at 5.94

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric,
## : neighborhood radius 2.06

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric,
## : reciprocal condition number 0

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric,
## : There are other near singularities as well. 4.2436
```

```r
loess_2_075 <- loess(vehicles ~ hour, data = traffic, degree = 2, span = 0.75)

# Predicting smoothed values at integer hours
hours <- seq(6, 18, by = 0.1)
pred_1_03 <- predict(loess_1_03, newdata = data.frame(hour = hours))
pred_1_075 <- predict(loess_1_075, newdata = data.frame(hour = hours))
pred_2_03 <- predict(loess_2_03, newdata = data.frame(hour = hours))
pred_2_075 <- predict(loess_2_075, newdata = data.frame(hour = hours))

# Creating scatter plot
plot(traffic$hour, traffic$vehicles, xlab = "Hour", ylab = "Vehicles", main = "Fitting with LOESS Smoot
     pch = 16, col = "black", ylim = c(150, 650))

# Adding LOESS smoother lines
lines(hours, pred_1_03, col = "red", lwd = 2, lty = 1)
lines(hours, pred_1_075, col = "green", lwd = 2, lty = 2)
lines(hours, pred_2_03, col = "orange", lwd = 2, lty = 3)
lines(hours, pred_2_075, col = "blue", lwd = 2, lty = 4)
legend("topleft",
       legend = c("Degree=1, Span=0.3",
                  "Degree=1, Span=0.75",
                  "Degree=2, Span=0.3",
                  "Degree=2, Span=0.75"),
       col = c("red", "green", "orange", "blue"),
       pch = c(NA, NA, NA, NA),
       lty = c(1, 2, 3, 4),
       lwd = c(2, 2, 2, 2))
```



Intepreting the behaviour

- From the LOESS smoothing plots, the span controls how much of the data is used in each local fit Smaller spans like, 0.3) produce a more wiggly curve that closely follows the data but risks overfitting, while larger spans like 0.75 create smoother trends that may underfit local variation. The degree determines the type of local regression. A degree of 1 fits local lines, while degree of 2 fits local quadratic functions (more flexible). Smaller spans and higher degrees increase sensitivity to local patterns, while larger spans and lower degrees prioritize smoothness. A span = 0.75 and degree = 2 appears to fit the data well.