Analytical and Simulation-Based Approaches to Local Linear Regression

Mini-project Epiphany term

Raul Unnithan, Supervisor: Dr John Einbeck

2025-04-29

1 Introduction

1.1 Motivation

In non-parametric regression, the aim is to estimate an unknown regression function, $m(x) = \mathbb{E}[Y \mid X = x]$, without pre-specifying a functional form. Among non-parametric methods, there are local polynomial estimators which approximate the regression function by fitting low-degree polynomials to subsets of the data, weighted by a kernel. This report looks at the local linear (LL) estimator, which balances bias and variance well and remains reliable near boundary regions.

The data examined in this report records planting density and yield in the production of white Spanish onions at two locations in South Australia (Purnong Landing and Virginia).

The aim of this report is to estimate the regression function linking onion density to expected yield and to quantify the uncertainty of this estimate using 95% pointwise confidence intervals (CIs).

Two methods are implemented. The first is an analytical approach based on asymptotic approximations, with plug-in estimates for bias and variance derived from the "rule-of-thumb" method. The second is a simulation-based method, where CIs are constructed empirically by repeatedly resampling the observed data and re-estimating the regression function using the semiparametric bootstrap.

Finally, these two approaches are compared in terms of visual behaviour and average interval width.

1.2 Exploratory Data Analysis

First, the necessary packages that were needed throughout the project were installed and loaded up.

```r
library(locpol)
library(SemiPar)
```

```
## Warning: package 'SemiPar' was built under R version 4.4.3
```

Next, the onions dataset was introduced and some initial exploratory data analysis was performed.

```r
data(onions)
str(onions)
```

```
## 'data.frame':    84 obs. of  3 variables:
##  $ dens    : num  23.5 26.2 27.8 32.9 33.3 ...
##  $ yield   : num  223 234 222 222 197 ...
##  $ location: int  0 0 0 0 0 0 0 0 0 0 ...
```

The three variables of the onions dataset were: density, yield and location. These refer to the areal density of plants (measured in plants per square metre), the onion yield (measured in grams per plant) and the indicator of location, where 0 = Purnong Landing and 1 = Virginia.

The location variable was removed as it is categorical and LL regression is designed for modelling smooth relationships with continuous predictors.

```
onions.data <- onions[, -3] # remove location
```

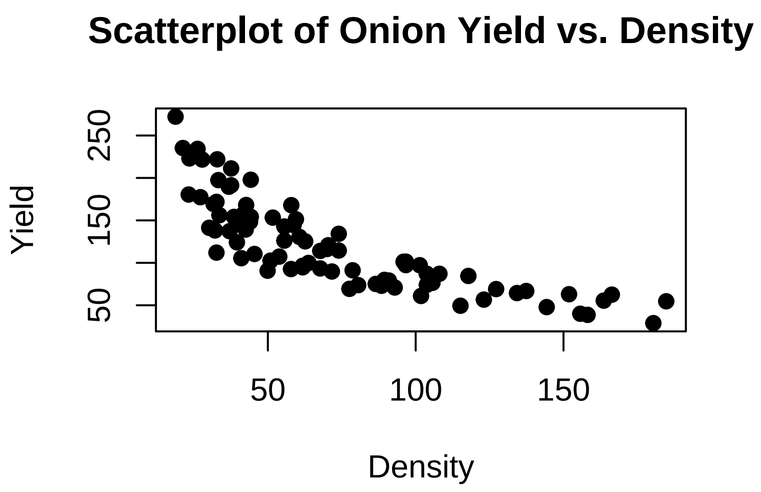
Missing data was then checked to confirm if any data needed to be imputed or rows with said missing data removed. However, the `any()` function confirmed there was no missing data.

```
any(is.na(onions.data))
```

```
## [1] FALSE
```

Next, a plot of `onions.data` was used to examine the relationship between onion density and yield.

```
plot(onions.data$dens, onions.data$yield, pch = 19,xlab = "Density",
     ylab = "Yield",main = "Scatterplot of Onion Yield vs. Density")
```



The above plot can be explained as follows: as planting density increases, each onion plant has access to less space, sunlight, and soil nutrients. This increased competition reduces the resources available for each plant to grow, resulting in a lower yield for each. This biological trade-off explains the observed negative correlation between onion yield and onion density.

2 Methodology

2.1 Set-Up

The formula for the LL estimator is:

$$\hat{m}_{LL}(x) = \frac{\sum_{i=1}^{n} v_i(x) y_i}{\sum_{i=1}^{n} v_i(x)}, \tag{1}$$

where:

$$v_i(x) = w_i(x)\left(S_{n,2} - (x_i - x)S_{n,1}\right), \qquad S_{n,j} = \sum_{i=1}^{n} w_i(x)(x_i - x)^j, \quad j = 0, 1, 2, \cdots. \tag{2}$$

Here, $w_i(x) = K_h(x_i - x)$ denotes the kernel weight applied to each observation, $S_{n,j}$ are the local moment sums, and $v_i(x)$ are the weights used to estimate the intercept of a locally weighted linear fit at point x.

The asymptotic bias and variance of the LL estimator, $\hat{m}_{LL}(x)$, are given by:

$$\text{Bias}[\hat{m}_{LL}(x)] = \frac{1}{2}h^2 m''(x)\mu_2 + o(h^2), \quad \text{Var}[\hat{m}_{LL}(x)] = \frac{\sigma^2}{nh} \cdot \frac{\nu_0}{f(x)} + o\left(\frac{1}{nh}\right), \tag{3}$$

where $m(x) = \mathbb{E}[Y \mid X = x]$ is the true regression function that we aim to estimate, and $m''(x)$ is its second derivative, capturing the local curvature of the function at point x. The bandwidth is denoted by h, and $f(x)$ is the probability density function (PDF) of the predictor variable X evaluated at point x.

The constants $\mu_2 = \int u^2 K(u)\,du$ and $\nu_0 = \int K^2(u)\,du$ are determined by the choice of kernel K, and represent the second and zeroth moment of the squared kernel, respectively. σ^2 denotes the conditional variance of the error term, and n is the sample size.

Now that the LL estimator has been set up with its bias and variance, its CI needed to be derived.

2.2 Deriving the Confidence Interval

The LL estimator is asymptotically normally distributed. This means the Lindeberg-Feller Central Limit Theorem applies:

$$\frac{\hat{m} - \mathbb{E}[\hat{m}]}{\sqrt{\text{Var}(\hat{m})}} \xrightarrow{d} \mathcal{N}(0, 1),$$

where $\xrightarrow{d}$ means "convergence in distribution" and $\hat{m}$ denotes the LL estimator, $\hat{m}_{LL}(x)$.

Let us ignore the $\mathcal{N}(0, 1)$ for now, and simplify the left-hand side as much as possible.

The bias formula, $\text{Bias}(\hat{m}) = \mathbb{E}(\hat{m}) - m$, can be rearranged and subbed in here for $\mathbb{E}(\hat{m})$. The variance, $\text{Var}(\hat{m})$, can also be plugged in directly (see the equations in 3 for these formulae). This gives the following expansion:

$$\frac{\hat{m} - \mathbb{E}[\hat{m}]}{\sqrt{\text{Var}(\hat{m})}} = \frac{\hat{m} - m - \text{Bias}(\hat{m})}{\sqrt{\frac{\sigma^2}{nh} \cdot \frac{\nu_0}{f(x)} + o\left(\frac{1}{nh}\right)}} = \frac{\hat{m} - m - \frac{h^2}{2}m''\mu_2 + o(h^2)}{\sqrt{\frac{\sigma^2}{nh} \cdot \frac{\nu_0}{f(x)} + o\left(\frac{1}{nh}\right)}}.$$

To remove the $o\left(\frac{1}{nh}\right)$ term, multiply the numerator and denominator by $\sqrt{nh}$ and simplify:

$$\frac{\sqrt{nh}}{\sqrt{nh}} \cdot \frac{\hat{m} - m - \frac{h^2}{2}m''\mu_2 + o(h^2)}{\sqrt{\frac{\sigma^2\nu_0}{nhf(x)} + o\left(\frac{1}{nh}\right)}} \quad \Rightarrow \quad \sqrt{nh} \cdot \frac{\hat{m} - m - \frac{h^2}{2}m''\mu_2 + o(h^2)}{\sqrt{\frac{\sigma^2\nu_0}{f(x)} + o(1)}}.$$

The term $o(1)$ in the denominator vanishes if $n \to \infty$, therefore it becomes irrelevant for the distributional statement being made. This means asymptotic normality remains true if it is omitted. Combining this with multiplying both sides by $\sqrt{\frac{\hat{\sigma}^2 \nu_0}{f(x)}}$ and using properties of the variance gives:

$$\sqrt{nh}\left(\hat{m} - m - \frac{h^2}{2} m'' \mu_2 + o(h^2)\right) \xrightarrow{d} \mathcal{N}\left(0, \frac{\sigma^2 \nu_0}{f(x)}\right). \tag{*}$$

Before proceeding, let us cover how $o(h^2)$ tends in distribution as $n \to \infty$. We will use the optimal bandwidth: $h_{opt} = cn^{-1/5}$, $c \in \mathbb{R}$. Remembering the $\sqrt{nh}$ outside the bracket, we have:

$$\sqrt{nh} \cdot o(h^2) = o(n^{1/2} h^{5/2}) = o\left(n^{1/2}\left(cn^{-1/5}\right)^{5/2}\right) = o(1),$$

and $o(1)$ tends to 0 as $n \to \infty$. Therefore, $o(h^2)$ can be omitted as well.

Now, continuing with equation (*), divide both sides by $\sqrt{nh}$ and use the properties of the variance:

$$\hat{m} - m - \frac{h^2}{2} m'' \mu_2 \xrightarrow{d} \mathcal{N}\left(0, \frac{\sigma^2 \nu_0}{nhf(x)}\right).$$

Given this asymptotic distribution, an approximate $(1-\alpha)\%$ CI for m, is:

$$\hat{m} - \frac{h^2}{2} m'' \mu_2 \pm z_{1-\alpha/2} \cdot \sqrt{\frac{\sigma^2 \nu_0}{f(x)\, nh}} \Rightarrow \hat{m} - \text{Bias}(\hat{m}) \pm z_{1-\alpha/2} \cdot \sqrt{\text{Var}(\hat{m})}, \tag{4}$$

where the constant $z_{1-\alpha/2}$ is an approximate normal quantile.

This final simplification follows directly from the expressions for the Variance and Bias of the LL estimator as defined in equation 3. It gives the form of the (bias-corrected) analytical CI.

We had now set up a good platform to starting building our CIs. However, some of the parameters could not be derived using exact values and instead they needed plug-in estimates.

2.3 Rule-of-Thumb Bandwidth

There exists a method to calculate these plug-in estimates using the "rule-of-thumb" bandwidth selector for LL regression. Under this approach, $\hat{m}(x)$ is estimated globally, using a polynomial of degree 4, as:

$$\check{m}(x) = \check{\alpha}_0 + \check{\alpha}_1 x + \check{\alpha}_2 x^2 + \check{\alpha}_3 x^3 + \check{\alpha}_4 x^4. \tag{5}$$

This means $m''(x)$ can be estimated using the second derivative of the fitted polynomial as:

$$\check{m}''(x) = 2\check{\alpha}_2 + 6\check{\alpha}_3 x + 12\check{\alpha}_4 x^2. \tag{6}$$

This method also estimates the error standard deviation, σ, from the fitted model's residuals using:

$$\check{\sigma}^2 = \frac{1}{n-5} \sum_{i=1}^{n} \left(y_i - \check{m}(x_i)\right)^2. \tag{7}$$

Finally, the above equations can be combined to estimate the "rule-of-thumb" bandwidth, h_{ROT}:

$$h_{\text{ROT}} = \left[\frac{\nu_0\, \check{\sigma}^2}{\mu_2^2 \sum_{i=1}^{n} \check{m}''(x_i)^2}\right]^{1/5} n^{-1/5}, \tag{8}$$

which is of the form $cn^{-1/5}$, like the optimal bandwidth used in the prior section.

The equations in this subsection now meant we could implement the analytical CI (see equation 4).

3 Code

3.1 Helper Functions

To implement both LL regression CI approaches, we first define several helper functions.

```r
my.kernel<-function(u){return(dnorm(u))}
Sn <- function(xdat, x, h, j){sum(my.kernel((xdat - x)/h)*(xdat - x)^j)}
vix <- function(xdat, x, h){
  my.kernel((xdat - x)/h)*(Sn(xdat, x, h, 2) - (xdat - x)*Sn(xdat, x, h, 1))}

my.ll.smoother <- function(xdat, ydat, xgrid = xdat, h){
  G <- length(xgrid)
  est <- rep(0, G)
  for (j in 1:G){
    est[j] <- sum(ydat*vix(xdat, xgrid[j], h))/sum(vix(xdat, xgrid[j], h))}
  return(est)}
```

The function `my.kernel()` returns the kernel weights used for local smoothing. In this report, a Gaussian kernel was predominantly used (see section 4 for further justification). `my.kernel()` implements $K(u)$ in the formula for μ_2 (see the explanation in equation 3 for this formula).

The function `Sn()` calculates the required moments for local polynomial regression and `vix()` computes the weights used in the LL estimation process. These functions implement the equations in 2.

The function `my.ll.smoother()` then brings `my.kernel()`, `Sn()` and `vix()` together to perform the LL estimates over the specified grid of evaluation points, `xgrid` (applying equation 1).

```r
m <- function(coefs, x) {coefs[1] + coefs[2]*x +
    coefs[3]*x^2 + coefs[4]*x^3 + coefs[5]*x^4}
m2 <- function(coefs, x) {2 * coefs[3] + 6 * coefs[4]*x + 12 * coefs[5]*x^2}

bias.ci <- function(h, m2, mu2) {(1/2) * (h^2) * m2 * mu2}
sd.ci <- function(n, h, sigma2, nu0, f) {
  variance <- (1 / (n * h)) * (sigma2) * (nu0 / f)
  return(variance^(1/2))}
```

The functions `m()` and `m2()` evaluate the fitted polynomial and its second derivative, respectively, and the latter was used in the estimation of bias. These functions implement equations 5 and 6, respectively.

The functions `bias()` and `sd()` compute the analytical bias and standard deviation required for constructing the bias-corrected analytical CIs. These derive the bias and the square root of the variance in equation 3. (Note: the square root was used for brevity in implementing equation 4.)

```r
est_density <- function(data, xgrid, h, kernel = "gauss") {
  my.kernel <- switch(kernel, "gauss" = function(x) dnorm(x),
                      "epa"   = function(x) { 3/4 * (1 - x^2) *
                          ifelse(abs(x) <= 1, 1, 0) },
                      "uni"   = function(x) { 0.5 * ifelse(abs(x) <= 1, 1, 0) })
  n <- length(data)
  est <- numeric(length(xgrid))
  denom <- n * h
  for (j in seq_along(xgrid)) {
    est[j] <- sum(my.kernel((data - xgrid[j]) / h)) / denom}
  return(est)}
```

Lastly, the function `est_density()` provides an estimate of the density of the predictor variable, $f(x)$. This was necessary in calculating the variance of the LL estimator.

3.2 Analytical Approach

The function `ana.ll()` was constructed to implement the analytical pointwise CI for the LL estimator.

```r
ana.ll <- function(data, xgrid, h = 1, kernel = "gauss", alpha = 0.05) {
  # kernel constants
  nu0 <- switch(kernel,"gauss" = 1 / (2 * sqrt(pi)),"epa" = 3 / 5,"uni" = 1 / 2)
  mu2 <- switch(kernel,"gauss" = 1,"epa" = 1 / 5,"uni" = 1 / 3)

  n <- nrow(data)
  x <- data[, 1]
  y <- data[, 2]

  model <- lm(y ~ poly(x, 4, raw = TRUE)) # Fit degree 4 polynomial
  coefs <- coef(model)
  mhat <- my.ll.smoother(xdat = x, ydat = y, xgrid = xgrid, h = h)
  mddash <- m2(coefs, xgrid)
  sigma2 <- sum((y - m(coefs, x))^2) / (n - 5)
  f <- est_density(x, xgrid, h, kernel)

  bias_val <- bias.ci(h, mddash, mu2)
  sd_val <- sd.ci(n, h, sigma2, nu0, f)

  est <- mhat - bias_val # Bias-corrected estimate
  z <- qnorm(1-alpha/2)
  lower_ci <- est - z * sd_val # Compute CI: (estimate - bias) - z * sd
  upper_ci <- est + z * sd_val # Compute CI: (estimate - bias) + z * sd

  ord <- order(xgrid) # Sort by xgrid to ensure lines() function plots properly
  xgrid_sorted <- xgrid[ord]
  est_sorted <- est[ord]
  lower_ci_sorted <- lower_ci[ord]
  upper_ci_sorted <- upper_ci[ord]
  return(list(xgrid = xgrid_sorted, est = est_sorted,
              lower = lower_ci_sorted, upper = upper_ci_sorted))}
```

The function begins by assigning the constants ν_0 and μ_2 based on the designated choice of kernel. These values are kernel-dependent and were used in the variance and bias calculations (see the equations in 3). The `switch()` statement allows for switching between Gaussian, Epanechnikov and Uniform kernels.

The LL estimate, $\hat{m}_{LL}(x)$, itself is then computed using a separate smoothing function `my.ll.smoother()`.

In line with the plug-in methodology described above, a global polynomial of degree 4 is fitted to the data. The coefficients from this fit are then used to compute plug-in estimates for the second derivative, $m''(x)$, and the residual variance, σ^2.

The PDF, $f(x)$, is then obtained using `est_density()`. This estimate is known as the kernel density estimate (KDE) and it is evaluated at each point in the grid. This allows the pointwise LL variance to use the density of the predictor variable.

All of the above then enables bias and variance for the LL estimator to then be computed.

The function finishes with sorting the evaluation grid and outputting the smoothed estimate along with the upper and lower confidence bounds, formatted for direct plotting using the `lines()` function.

3.3 Simulated Approach

The function `sim.ll()` was constructed to implement the simulated pointwise CI for LL regression using semiparametric bootstrap.

```r
sim.ll <- function(data, xgrid, h = 1, B = 300, alpha = 0.05) {
  n <- nrow(data)
  x <- data[, 1]
  y <- data[, 2]
  mhat <- my.ll.smoother(xdat = x, ydat = y, xgrid = xgrid, h = h)
  mhat_at_x <- my.ll.smoother(xdat = x, ydat = y, xgrid = x, h = h)
  residuals <- y - mhat_at_x # to be resampled

  # Initialise bootstrap matrix
  boots <- matrix(NA, nrow = B, ncol = length(xgrid))
  for (b in 1:B) {
    resampled_resid <- sample(residuals, replace = TRUE)
    y_star <- mhat_at_x + resampled_resid # new prediction
    boots[b, ] <- my.ll.smoother(xdat = x, ydat = y_star, xgrid = xgrid, h = h)}
  lower_ci <- apply(boots, 2, quantile, probs = alpha / 2)
  upper_ci <- apply(boots, 2, quantile, probs = 1 - alpha / 2)
  ord <- order(xgrid)
  return(list(xgrid = xgrid[ord], est = mhat[ord],
    lower = lower_ci[ord], upper = upper_ci[ord]))}
```

The function first computes the LL regression estimate $\hat{m}(x)$ over the evaluation grid `xgrid` using `my.ll.smoother()`. Fitted values are then calculated at the observed x-values, and the residuals are obtained as: $\hat{\varepsilon}_i = y_i - \hat{m}(x_i)$.

Next, a matrix is initialised to store the B bootstrap simulations of the smoothed curve. For each bootstrap iteration, residuals are resampled with replacement from the original residuals. Then, a new pseudo-response is constructed as follows:

$$y_i^* = \hat{m}(x_i) + \hat{\varepsilon}_i^*.$$

Then, a new local linear fit is computed on the synthetic dataset (x_i, y_i^*) and the fitted values at `xgrid` are stored in the bootstrap matrix. Finally, for each evaluation point $x \in$ `xgrid`, the `apply()` function calculates the quantiles column-wise, i.e. across bootstrap samples for each evaluation point to form a two-sided, $1-\alpha$, CI:

$$\left[\hat{m}^{(\alpha/2)}(x), \hat{m}^{(1-\alpha/2)}(x)\right].$$

4 Analysis

LL regression was implemented here using a Gaussian kernel because it offers smooth estimates and stable performance at the interval boundaries compared to other kernels such as Epanechikov and Uniform.

R has an in-built function, `thumbBw()`, from the package `locpol` that implements equation 8, returning a suitable bandwidth. Here is how it was applied on the `onions.data`:

```r
h_rot <- thumbBw(onions.data$dens, onions.data$yield, deg = 1, kernel = gaussK)
h_rot
```

```
## [1] 3.293061
```

This procedure can also be implemented manually as follows:

```r
thumbBw_manual <- function(x, y, kernel = "gauss") {
  n <- length(x)
  nu0 <- switch(kernel, "gauss" = 1 / (2 * sqrt(pi)), "epa" = 3 / 5, "uni" = 1 / 2)
  mu2 <- switch(kernel, "gauss" = 1, "epa"   = 1 / 5, "uni" = 1 / 3)
  poly_fit <- lm(y ~ poly(x, degree = 4, raw = TRUE))
  coefs <- coef(poly_fit)
  sigma2 <- sum((y - m(coefs, x))^2) / (n - 5)
  sum_m_double_sq <- sum(m2(coefs, x)^2)
  numerator <- nu0 * sigma2
  denominator <- mu2^2 * sum_m_double_sq
  h_rot <- (numerator / denominator)^(1/5) * n^(-1/5)
  return(h_rot)}
h_rot_manual <- thumbBw_manual(onions.data$dens, onions.data$yield, kernel = "gauss")
h_rot_manual
```

```
## [1] 1.374277
```

Although both the manual and package-based procedures aim to implement the same bandwidth, they differ in how key quantities are estimated. `thumbBw()` estimates the bias and variance using local polynomial fits and internal kernel constants via `cteNuK()` (see section 6.1 for the full structure of the in-built function for `thumbBw`). However, `thumbBw_manual` estimates the bias and variance using a global, degree 4, polynomial fit and pre-specified kernel constants. These methodological differences explain the discrepancy in the resulting bandwidth values.

Note: the manual bandwidth was used for these approaches (see section 6.2 for the use of `thumbBw()`).

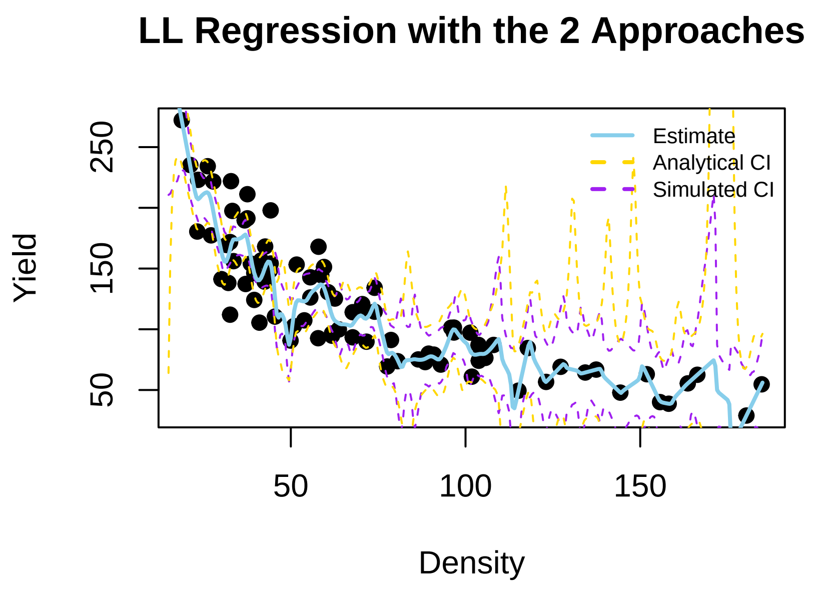
The summary of the data was then used to guide the selection of the evaluation grid, `x_grid`, when constructing the CIs. The grid was chosen to span a range slightly beyond the observed values of `onions.data$dens`, covering approximately the minimum to the maximum observed onion densities.

```r
summary(onions.data)
```

```
##       dens            yield
##  Min.   : 18.78   Min.   : 28.96
##  1st Qu.: 39.54   1st Qu.: 78.46
##  Median : 61.78   Median :108.90
##  Mean   : 73.33   Mean   :119.70
##  3rd Qu.: 97.86   3rd Qu.:153.47
##  Max.   :184.75   Max.   :272.15
```

```
xgrid <- seq(15, 185, by = 0.5) # now, we were ready to plot:
```

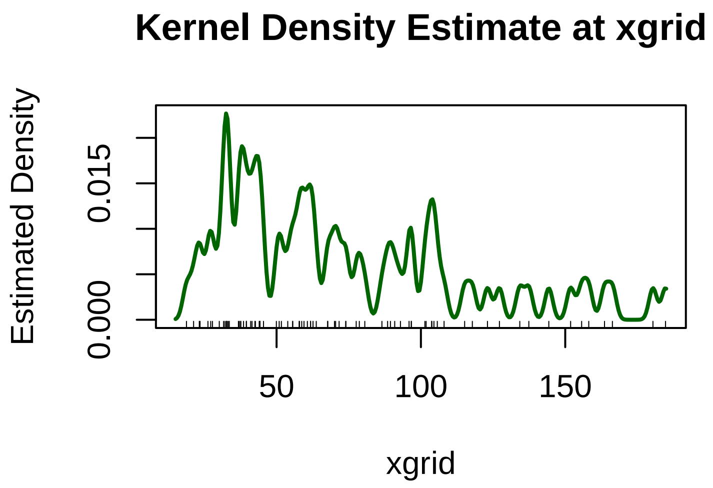
```
res_analytic <- ana.ll(onions.data, xgrid = xgrid, h = h_rot_manual)
res_bootstrap <- sim.ll(onions.data, xgrid = xgrid, h = h_rot_manual)
plot(onions.data$dens, onions.data$yield, pch = 19,xlab = "Density",
     ylab = "Yield", main = "LL Regression with the 2 Approaches")

# Simulated Approach
lines(res_bootstrap$xgrid, res_bootstrap$lower, col = "purple", lty = 2)
lines(res_bootstrap$xgrid, res_bootstrap$upper, col = "purple", lty = 2)
# Analytical Approach
lines(res_analytic$xgrid, res_analytic$est, col = "skyblue", lwd = 2)
lines(res_analytic$xgrid, res_analytic$lower, col = "gold", lty = 2)
lines(res_analytic$xgrid, res_analytic$upper, col = "gold", lty = 2)
legend("topright", legend = c("Estimate", "Analytical CI", "Simulated CI"), cex = 0.7,
       col = c("skyblue", "gold", "purple"), lwd = 2, lty = c(1, 2, 2), bty = "n")
```



The regression estimate declines as density increases, as expected. However, mild local overfitting suggests the bandwidth is slightly too small. A larger bandwidth would stabilise the estimate by reducing noise and returning narrower, more reliable CIs, particularly in regions with more observations (this is confirmed by the plot in section 6.2 using `thumbBw()`). Despite this, the overall trend is clear. The simulated CIs widen, as expected, in areas of low density, reflecting increased variance due to data scarcity. In contrast, the analytical CIs widen more abruptly due to their dependence on the reciprocal of the KDE function $\hat{f}(x)$.

As density approaches zero, variance inflates and the analytical CIs become jagged. This is confirmed by the following KDE plot:

```r
f <- est_density(onions.data$dens, xgrid, h_rot_manual, kernel="gauss")
plot(xgrid, f, type = "l", col = "darkgreen", lwd = 2,
     ylab = "Estimated Density", main = "Kernel Density Estimate at xgrid")
rug(onions.data$dens, col = "black") # displays where observations occurred in the grid
```



This issue was also in the Epanechikov and Uniform KDEs (see sections 6.3 and 6.4 for these plots, respectively). However, it can be mitigated by capping the KDE to a small value above zero to avoid numerical instabilities (see section 6.5 for an adjusted analytical function and plot for the Gaussian kernel).

Further code was also ran to compare the average widths of CIs for each approach:

```r
width_analytic <- mean(res_analytic$upper - res_analytic$lower)
width_bootstrap <- mean(res_bootstrap$upper - res_bootstrap$lower)
interval_widths <- data.frame(Method = c("Analytical Approach", "Simulated Approach"),
                              Average_Width = c(width_analytic, width_bootstrap))
interval_widths
```

```
##                Method Average_Width
## 1 Analytical Approach     432.51003
## 2  Simulated Approach      62.22027
```

This table furthers the insights from the LL regression plot, i.e. that the analytical approach produced wider intervals, by quantifying this uncertainty.

5 Conclusion

Overall, the comparison demonstrates that the analytical CIs overstates uncertainty in regions with few observations, while the simulated intervals provide a more stable and interpretable alternative in practice. Both CIs appear to be quite jagged which comes from the small bandwidth used. It is also important to not use a bandwidth that is too large as this leads to oversmoothing (see section 6.6 for such a plot).

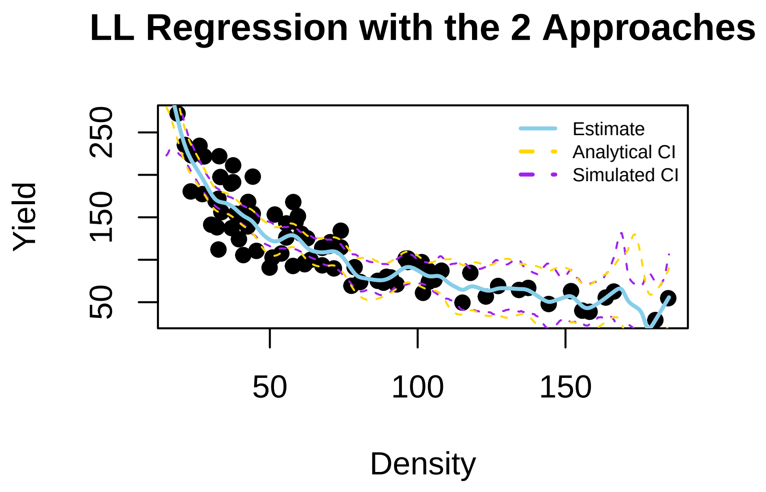
6 Appendix

6.1 Build of the `thumbBw` Function

```r
thumbBw_automatic <- function (x, y, deg, kernel, weig = rep(1, length(y))){
  k <- 3
  rd <- compDerEst(x, y, deg, weig)
  denom <- sum(rd$der^2)
  numer <- mean(rd$res^2)
  cte <- cteNuK(0, deg, kernel, lower = dom(kernel)[[1]], upper = dom(kernel)[[2]],
      subdivisions = 100)
  res <- cte * (numer/denom)^(1/(2 * deg + k))
  return(res)}
```

6.2 Automatic Implementation of the "Rule-of-Thumb" Bandwidth

```r
my.kernel<-function(u){return(dnorm(u))}
res_analytic <- ana.ll(onions.data, xgrid = xgrid, h = h_rot)
res_bootstrap <- sim.ll(onions.data, xgrid = xgrid, h = h_rot)
plot(onions.data$dens, onions.data$yield, pch = 19,xlab = "Density",
     ylab = "Yield", main = "LL Regression with the 2 Approaches")
# Simulated Approach
lines(res_bootstrap$xgrid, res_bootstrap$lower, col = "purple", lty = 2)
lines(res_bootstrap$xgrid, res_bootstrap$upper, col = "purple", lty = 2)
# Analytical Approach
lines(res_analytic$xgrid, res_analytic$est, col = "skyblue", lwd = 2)
lines(res_analytic$xgrid, res_analytic$lower, col = "gold", lty = 2)
lines(res_analytic$xgrid, res_analytic$upper, col = "gold", lty = 2)
legend("topright", legend = c("Estimate", "Analytical CI", "Simulated CI"), cex = 0.6,
       col = c("skyblue", "gold", "purple"), lwd = 2, lty = c(1, 2, 2), bty = "n")
```



Here is the average interval width for the analytical and simulation-based approaches using the `thumbBw()` function bandwidth:

```
width_analytic <- mean(res_analytic$upper - res_analytic$lower)
width_bootstrap <- mean(res_bootstrap$upper - res_bootstrap$lower)

interval_widths <- data.frame(Method = c("Analytical Approach", "Simulated Approach"),
                              Average_Width = c(width_analytic, width_bootstrap))
interval_widths
```

```
##                Method Average_Width
## 1 Analytical Approach      50.79783
## 2  Simulated Approach      46.68310
```

As expected, the average interval width significantly improves as bandwidth increases.

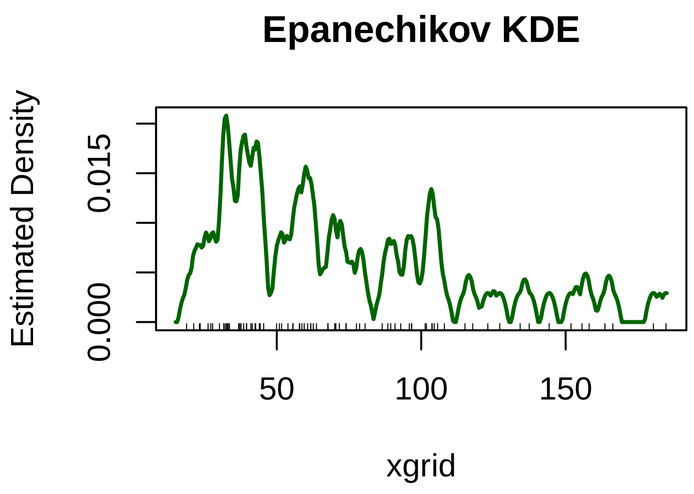
6.3 Epanechikov Kernel Densities

Nothing much changes here, just specifying the Epanechikov Kernel and changing the `"gauss"` argument to `"epa"` in `thumbBw_manual`.

```
my.kernel<-function(u){u <- 3/4*(1-u^2)*ifelse(abs(u)<=1,1,0)}

h_rot_epa_manual <- thumbBw_manual(onions.data$dens, onions.data$yield, kernel = "epa")
h_rot_epa_manual
```

```
## [1] 3.042381
```

```
f <- est_density(onions.data$dens, xgrid, h_rot_epa_manual, kernel="epa")
plot(xgrid, f, type = "l", col = "darkgreen", lwd = 2,
     ylab = "Estimated Density", main = "Epanechikov KDE")
rug(onions.data$dens, col = "black")
```



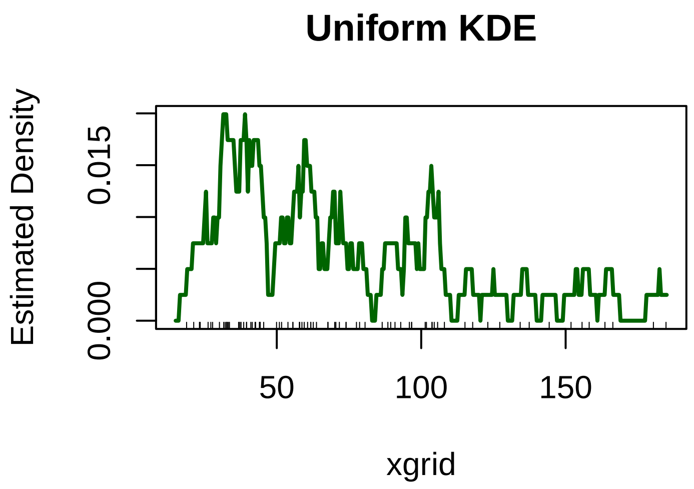
6.4 Uniform Kernel Densities

Now, the same changes as for Epanechikov but for the Uniform Kernel.

```
# Uniform
my.kernel<-function(u){u <- 0.5*ifelse(abs(u)<=1,1,0)}

h_rot_uni_manual <- thumbBw_manual(onions.data$dens, onions.data$yield, kernel = "uni")
h_rot_uni_manual
```

```
## [1] 2.391321
```

```
f <- est_density(onions.data$dens, xgrid, h_rot_uni_manual, kernel="uni")
plot(xgrid, f, type = "l", col = "darkgreen", lwd = 2,
     ylab = "Estimated Density", main = "Uniform KDE")
rug(onions.data$dens, col = "black") # displays where observations occur in xgrid
```



6.5 Adjusted Analytical LL Function

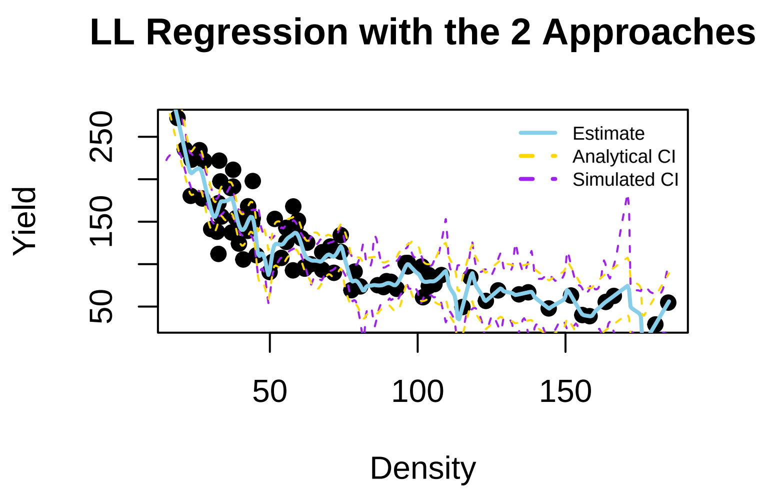
Here is the analytical function with the PDF capped to prevent small values from inflating the LL variance:

```
my.kernel<-function(u){return(dnorm(u))}

ana.adj.ll <- function(data, xgrid, h = 1, kernel = "gauss", alpha = 0.05) {
  # kernel constants
  nu0 <- switch(kernel,"gauss" = 1 / (2 * sqrt(pi)),"epa" = 3 / 5,"uni" = 1 / 2)
  mu2 <- switch(kernel,"gauss" = 1,"epa" = 1 / 5,"uni" = 1 / 3)

  n <- nrow(data)
  x <- data[, 1]
  y <- data[, 2]
```

```r
  model <- lm(y ~ poly(x, 4, raw = TRUE)) # Fit degree 4 polynomial
  coefs <- coef(model)
  mhat <- my.ll.smoother(xdat = x, ydat = y, xgrid = xgrid, h = h)
  mddash <- m2(coefs, xgrid)
  sigma2 <- sum((y - m(coefs, x))^2) / (n - 5)
  f <- est_density(x, xgrid, h, kernel)
  f <- pmax(f, 0.005)  # Prevents small KDEs from inflating LL variance

  bias_val <- bias.ci(h, mddash, mu2)
  sd_val <- sd.ci(n, h, sigma2, nu0, f)

  est <- mhat - bias_val # Bias-corrected estimate
  z <- qnorm(1-alpha/2)
  lower_ci <- est - z * sd_val # Compute CI: (estimate - bias) - z * sd
  upper_ci <- est + z * sd_val # Compute CI: (estimate - bias) + z * sd

  ord <- order(xgrid) # Sort by xgrid to ensure lines() function plots properly
  xgrid_sorted <- xgrid[ord]
  est_sorted <- est[ord]
  lower_ci_sorted <- lower_ci[ord]
  upper_ci_sorted <- upper_ci[ord]
  return(list(xgrid = xgrid_sorted, est = est_sorted,
              lower = lower_ci_sorted, upper = upper_ci_sorted))}
```

Here is its implementation:

```r
res_analytic <- ana.adj.ll(onions.data, xgrid = xgrid, h = h_rot_manual)
res_bootstrap <- sim.ll(onions.data, xgrid = xgrid, h = h_rot_manual)

plot(onions.data$dens, onions.data$yield, pch = 19,xlab = "Density",
     ylab = "Yield", main = "LL Regression with the 2 Approaches")

# Simulated Approach
lines(res_bootstrap$xgrid, res_bootstrap$lower, col = "purple", lty = 2)
lines(res_bootstrap$xgrid, res_bootstrap$upper, col = "purple", lty = 2)

# Analytical Approach
lines(res_analytic$xgrid, res_analytic$est, col = "skyblue", lwd = 2)
lines(res_analytic$xgrid, res_analytic$lower, col = "gold", lty = 2)
lines(res_analytic$xgrid, res_analytic$upper, col = "gold", lty = 2)
legend("topright", legend = c("Estimate", "Analytical CI", "Simulated CI"), cex = 0.6,
       col = c("skyblue", "gold", "purple"), lwd = 2, lty = c(1, 2, 2), bty = "n")
```



```
width_analytic <- mean(res_analytic$upper - res_analytic$lower)
width_bootstrap <- mean(res_bootstrap$upper - res_bootstrap$lower)

interval_widths <- data.frame(Method = c("Analytical Approach", "Simulated Approach"),
                              Average_Width = c(width_analytic, width_bootstrap))
interval_widths
```

```
##                Method Average_Width
## 1 Analytical Approach      58.66063
## 2  Simulated Approach      61.12223
```

Here, you can clearly see how the interval width has decreased for the analytical approach and it is now smaller than the simulated approach.

6.6 Oversmoothed Function

This uses the package `KernSmooth`.

```
library(KernSmooth)
```

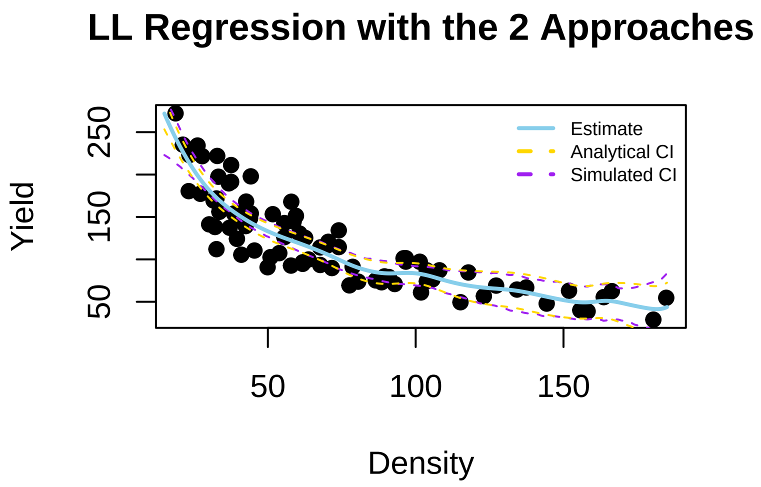
```
## Warning: package 'KernSmooth' was built under R version 4.4.3
```

```
## KernSmooth 2.23 loaded
## Copyright M. P. Wand 1997-2009
```

```
h_dpill <- dpill(onions.data$dens, onions.data$yield)
h_dpill
```

```
## [1] 7.921906
```

```r
my.kernel<-function(u){return(dnorm(u))}

res_analytic <- ana.ll(onions.data, xgrid = xgrid, h = h_dpill)
res_bootstrap <- sim.ll(onions.data, xgrid = xgrid, h = h_dpill)

plot(onions.data$dens, onions.data$yield, pch = 19,xlab = "Density",
     ylab = "Yield",main = "LL Regression with the 2 Approaches")

# Simulated Approach
lines(res_bootstrap$xgrid, res_bootstrap$lower, col = "purple", lty = 2)
lines(res_bootstrap$xgrid, res_bootstrap$upper, col = "purple", lty = 2)

# Analytical Approach
lines(res_analytic$xgrid, res_analytic$est, col = "skyblue", lwd = 2)
lines(res_analytic$xgrid, res_analytic$lower, col = "gold", lty = 2)
lines(res_analytic$xgrid, res_analytic$upper, col = "gold", lty = 2)
legend("topright", legend = c("Estimate", "Analytical CI", "Simulated CI"), cex = 0.6,
       col = c("skyblue", "gold", "purple"), lwd = 2, lty = c(1, 2, 2), bty = "n")
```



```r
width_analytic <- mean(res_analytic$upper - res_analytic$lower)
width_bootstrap <- mean(res_bootstrap$upper - res_bootstrap$lower)

interval_widths <- data.frame(Method = c("Analytical Approach", "Simulated Approach"),
                              Average_Width = c(width_analytic, width_bootstrap))
interval_widths
```

```
##                Method Average_Width
## 1 Analytical Approach      31.00706
## 2  Simulated Approach      31.68577
```

This is the lowest interval width and although it captures the general trend, the oversmoothing fails to capture important local patterns in the data.