# Microeconometrics Task - Problem Set 3

Ricardo Semião e Castro

2024-06-06

## 1 Setup

The packages used were:

```
library(nprobust)
library(gt)
library(furrr)
library(tidyverse)

set.seed(0306152529)
theme_set(theme_bw())
plan(multisession, workers = 7)
```

Lets define a function to plot the results from the models in this questions:

```
plot_lp <- function(x, tau_col = "tau.us", se_col = "se.rb") {
  ggplot(data, aes(x1, f)) +
  geom_line() +
  geom_line(aes(eval, .data[[tau_col]]), x, linetype = "dashed") +
  geom_ribbon(
    aes(
      eval, .data[[tau_col]],
      ymin = .data[[tau_col]] - .data[[se_col]],
      ymax = .data[[tau_col]] + .data[[se_col]]
    ),
    x, linetype = "dashed", fill = NA, color = "blue"
  )
}
```

## 2 Question 1 and 2

We can create the simulated dataset with the following function:

```
sim_data <- function(n, beta1, beta2) {
  e <- replicate(2, rnorm(n))

  x <- mvtnorm::rmvnorm(n, sigma = matrix(c(1, 0.9, 0.9, 1), 2, 2)) %>%
    apply(2, pnorm)

  f <- sin(beta1 * x[,1])

  y <- cbind(f + e[,1], f + beta2 * x[,2] + e[,2])
```

```r
  cbind(x, y, f) %>%
    `colnames<-`(c("x1", "x2", "y1", "y2", "f")) %>%
    .[order(.[, "x1"]), ]
}
```

We can now define the parameters and create the simulated dataset:

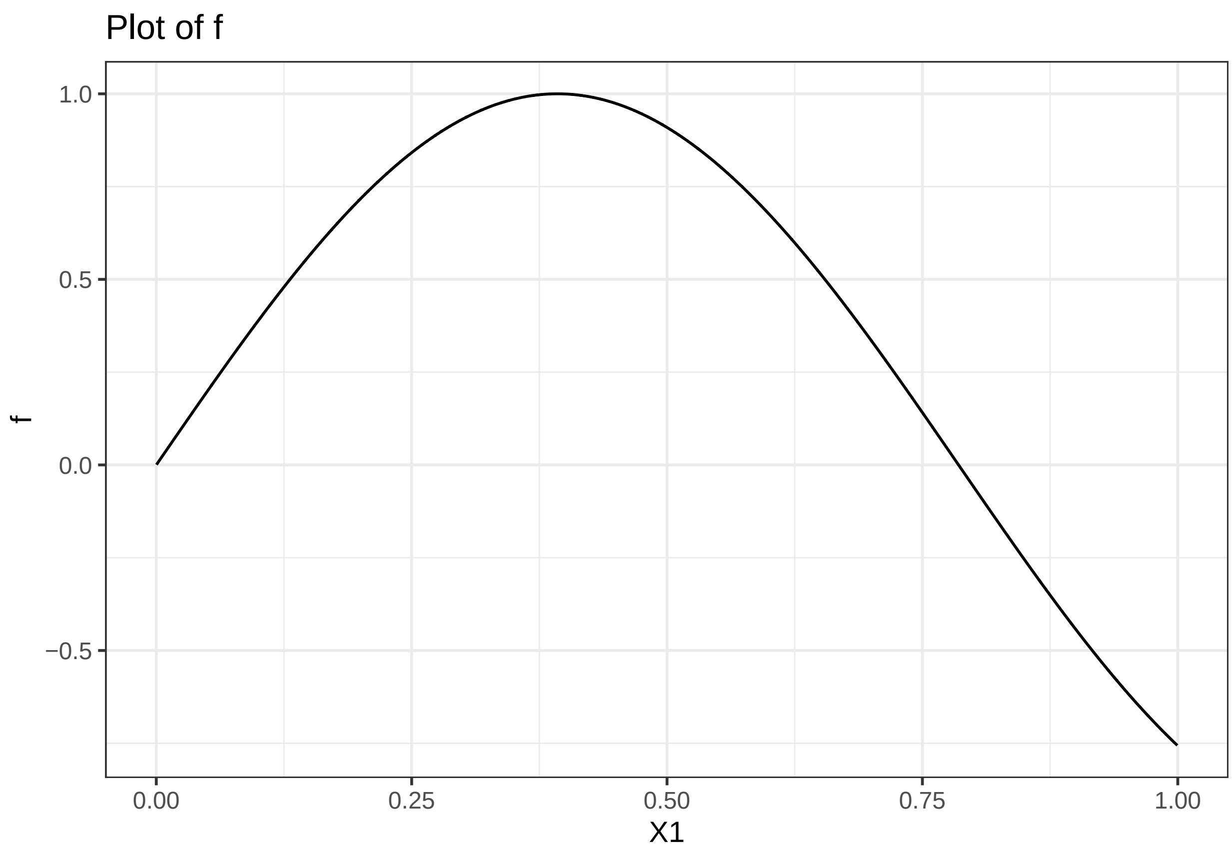
```r
n <- 10000

beta1 <- 4
beta2 <- 2

data <- sim_data(n, beta1, beta2)
```

And finally, plot the function `sin(beta1 * x1)`:

```r
ggplot(data, aes(x1, f)) +
  geom_line() +
  labs(title = "Plot of f", y = "f", x = "X1")
```




### 2.1 Question 3 and 4

We can use the `lprobust` package/function to estimate a local polinomial on $Y_1$, using the data from $X_1$.

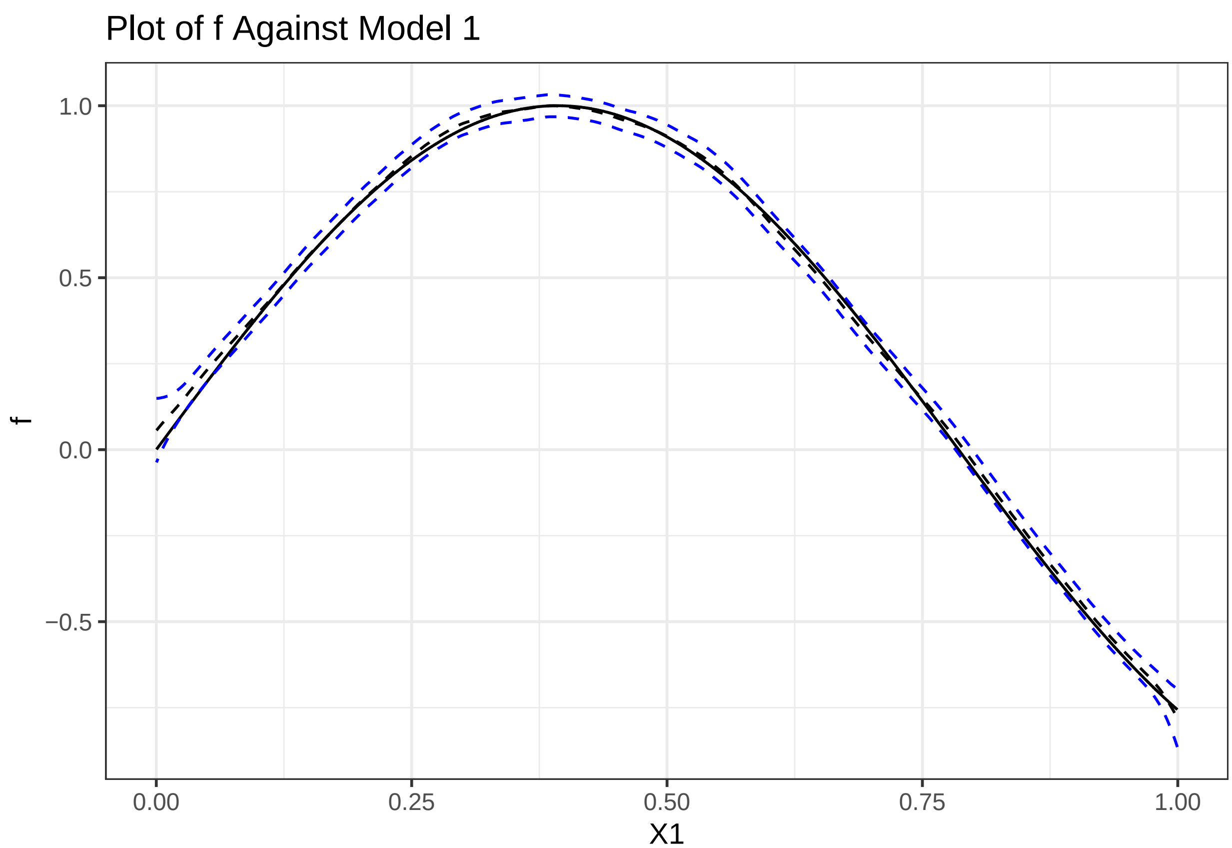
The specification of the model considers:

- `n` points of evaluation, related to the parameter `neval`.

- Polinomial order of 0, related to the parameter `p`. A level of zero is related to a "simple" local linear regression, as wanted.
- Derivative order of 0, related to the parameter `deriv`. To estimate a regression function, as wanted.
- Bandwith $h$ selected using `lpbwselect`, following the method of Calonico, Cattaneo and Farrell (2018) – "second-generation DPI implementation of IMSE-optimal bandwidth". The same was used for the bias correcting bandwith $b$.
- Kernel: Epanechnikov kernel.

```
mod1 <- lprobust(data[, "y1"], data[, "x1"], neval = n)

plot_lp(mod1$Estimate) +
  labs(title = "Plot of f Against Model 1", y = "f", x = "X1")
```




Besides the visualization, we can get statistics for the distance between the function and its estimated counterpart. There are several options, for example:

```
agg_funs <- list(
  `Mean abs.` = mean,
  `Mean sqr.` = \(x) mean(x^2),
  `Max abs.` = max,
  `Median abs.` = median
)
```

Consider the function below to extract these measures of difference from the model. Note that we might also be interested in considering:

- The difference between the true and the estimate.
- An lower bound, using the lowest difference among the whole confidence interval, at each `x`.

- An upper bound, using the highest difference among the whole confidence interval, at each `x`.

```
diff_lp <- function(x, tau_col = "tau.us", se_col = "se.rb") {
  n <- nrow(x)

  values <- cbind(
      x[, tau_col] - x[, se_col] - data[, "f"],
      x[, tau_col] - data[, "f"],
      x[, tau_col] + x[, se_col] - data[, "f"]
    )

  imap_dfr(agg_funs, function(fun, name) {
    list(
      Measure = name,
      Estimate = fun(abs(values[, 2])) / n,
      Lower = fun(apply(values, 1, \(x) min(abs(x)))) / n,
      Upper = fun(apply(values, 1, \(x) max(abs(x)))) / n
    )
  })
}
```

Lets apply it to our model 1:

```
diff_lp(mod1$Estimate) %>%
    gt() %>%
    fmt_scientific(-1, decimals = 2)
```

| Measure | Estimate | Lower | Upper |
|---|---|---|---|
| Mean abs. | $1.19 \times 10^{-6}$ | $9.18 \times 10^{-7}$ | $4.65 \times 10^{-6}$ |
| Mean sqr. | $2.25 \times 10^{-8}$ | $1.09 \times 10^{-8}$ | $2.34 \times 10^{-7}$ |
| Max abs. | $5.60 \times 10^{-6}$ | $3.77 \times 10^{-6}$ | $1.49 \times 10^{-5}$ |
| Median abs. | $1.03 \times 10^{-6}$ | $9.32 \times 10^{-7}$ | $4.37 \times 10^{-6}$ |

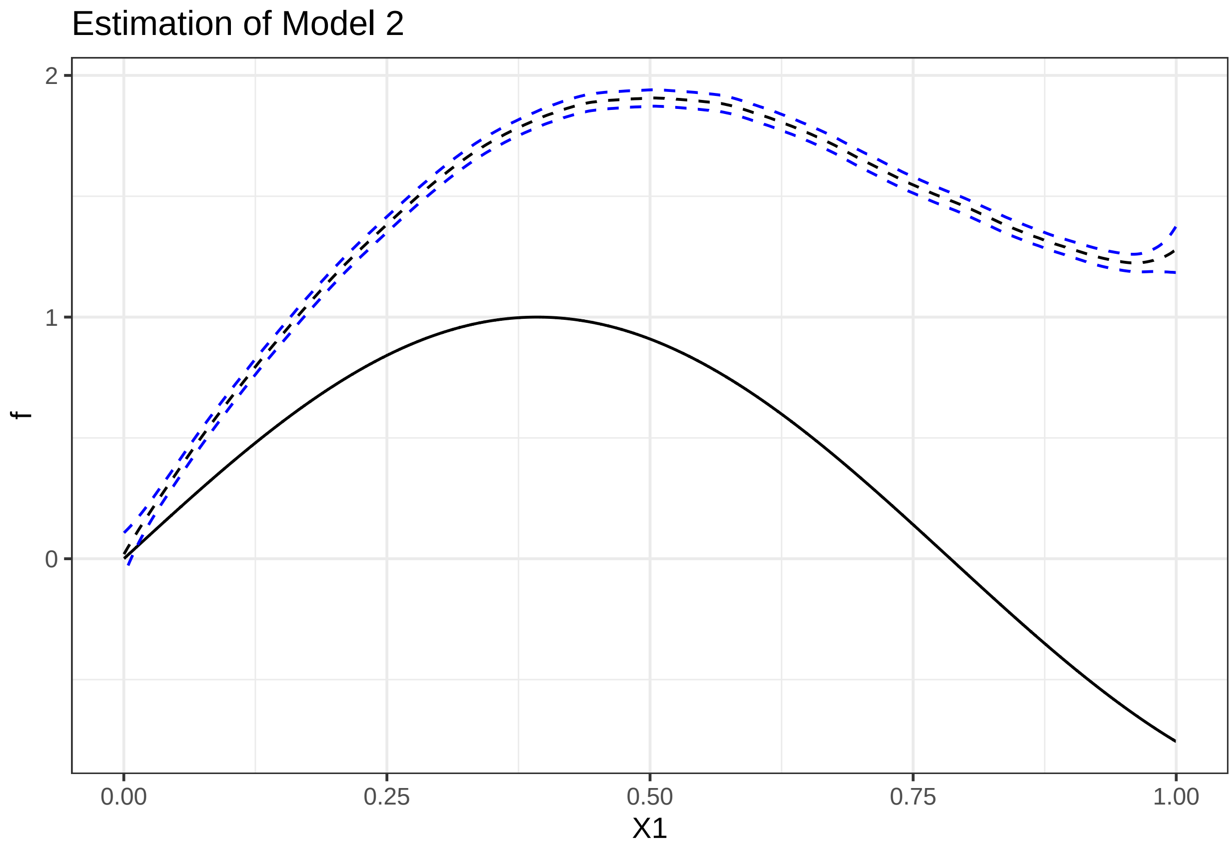
We can see that the values are very similar, which is expected, as indeed, $Y_1$ is explained only by $X_1$ and some random variation ($\epsilon_1$)

## 2.2 Question 5 and 6

The model specification is the same, but now we use $Y_2$ against $X_1$.

```
mod2 <- lprobust(data[, "y2"], data[, "x1"], neval = n)

plot_lp(mod2$Estimate) +
  labs(title = "Estimation of Model 2", y = "f", x = "X1")
```




And the differences statistics:

```r
diff_lp(mod2$Estimate) %>%
    gt() %>%
    fmt_scientific(-1, decimals = 2)
```

| Measure | Estimate | Lower | Upper |
|---|---|---|---|
| Mean abs. | $9.90 \times 10^{-5}$ | $9.56 \times 10^{-5}$ | $1.03 \times 10^{-4}$ |
| Mean sqr. | $1.26 \times 10^{-4}$ | $1.19 \times 10^{-4}$ | $1.33 \times 10^{-4}$ |
| Max abs. | $2.03 \times 10^{-4}$ | $1.94 \times 10^{-4}$ | $2.13 \times 10^{-4}$ |
| Median abs. | $9.91 \times 10^{-5}$ | $9.57 \times 10^{-5}$ | $1.02 \times 10^{-4}$ |

We can see that the values are very different, which is expected, as indeed $Y_2$ is not explained only by $X_1$ through $f$, the estimated function is also capturing the effect of $\beta_2 X_2$ in $Y_2$ through its correlation with $X_1$. The estimation is indeed similar to a combination of a sin curve and a linear trend.

## 2.3 Question 7, 8, and 9

Following what was stated above, we need to cleanse the relation that $X_1$ has on $Y_2$ through $\beta_2 X_2$. For that, we will:

- Find the (non-linear) relation between $X_2$ and $X_1$, and between $Y_2$ and $X_1$.
- Then with the unexplained parts of such relations (the residuals), we can find the relation $Y_2 \sim X_2$.
- We can use that to "cleanse" $Y_2$ from the effect from $X_2$.
- Lastly, we can use that adjusted $Y_2$ to find a better estimation for $f$.

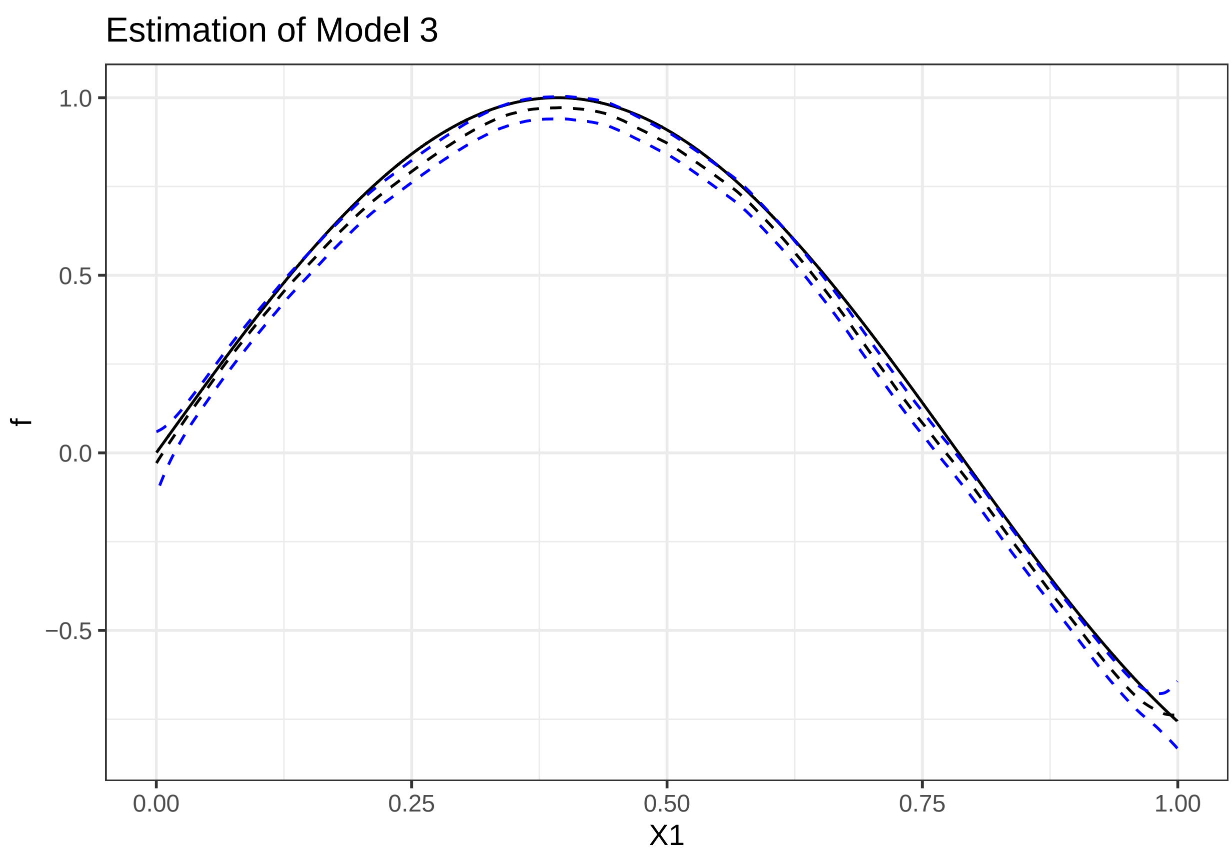
```r
mod3_x2 <- lprobust(data[, "x2"], data[, "x1"], neval = n)
x2_res <- data[, "x2"] - mod3_x2$Estimate[, "tau.us"]

mod3_y2 <-  lprobust(data[, "y2"], data[, "x1"], neval = n)
y2_res <- data[, "y2"] - mod3_y2$Estimate[, "tau.us"]

mod3_res <- lm(y2_res ~ 0 + x2_res)

y2_adj <- data[, "y2"] - data[, "x2"] * mod3_res$coefficients[1]
mod3 <- lprobust(y2_adj, data[, "x1"], neval = n)

plot_lp(mod3$Estimate) +
  labs(title = "Estimation of Model 3", y = "f", x = "X1")
```




And the differences statistics:

```r
diff_lp(mod3$Estimate) %>%
    gt() %>%
    fmt_scientific(-1, decimals = 2)
```

| Measure | Estimate | Lower | Upper |
|---|---|---|---|
| Mean abs. | $3.82 \times 10^{-6}$ | $1.06 \times 10^{-6}$ | $7.21 \times 10^{-6}$ |
| Mean sqr. | $1.61 \times 10^{-7}$ | $1.85 \times 10^{-8}$ | $5.34 \times 10^{-7}$ |
| Max abs. | $7.09 \times 10^{-6}$ | $3.79 \times 10^{-6}$ | $1.18 \times 10^{-5}$ |

| | | | |
|---|---|---|---|
| Median abs. | $3.83 \times 10^{-6}$ | $8.78 \times 10^{-7}$ | $7.17 \times 10^{-6}$ |

We can see that the values are very similar. This is a sign that our logic of cleansing the effect of $X_2$ was actually right, and we got a estimation only on the effect of $X_1$ trough $f$.

The estimated $beta_2$ can be seen below. It is decently close to 2, as it should, following the Frisch-Waugh-Lovell-like result that was used in its estimation.

```
summary(mod3_res)$coefficients["x2_res", ]
```

```
##      Estimate    Std. Error       t value      Pr(>|t|)
##  2.051023e+00  7.580837e-02  2.705537e+01 1.210696e-155
```

## 2.4 Extra: Monte Carlo

As an extra exercise, I've created a possible Monte-Carlo solution. I'll use the `furrr` package for parallel computing. We just need to estimate each of the three models in a loop, saving the results.

As this was too computationally expensive, I did not run the full loop to get the results. Still, I did check if the code would be able to run correctly.

```
n_iter <- 1000

results <- future_map(seq_len(n_iter), function(iter) {
  data <- sim_data(n, beta1, beta2)

  mod1 <- lprobust(data[, "y1"], data[, "x1"], neval = n)

  mod2 <- lprobust(data[, "y2"], data[, "x1"], neval = n)

  mod3_param <- lprobust(data[, "x2"], data[, "x1"], neval = n)
  x2_res <- data[, "x2"] - mod3_param$Estimate[, "tau.us"]
  mod3_nparam <-  lprobust(data[, "y2"], data[, "x1"], neval = n)
  y2_res <- data[, "y2"] - mod3_nparam$Estimate[, "tau.us"]
  mod3_res <- lm(y2_res ~ -1 + x2_res, neval = n)

  list(
    mod1 = mod1$Estimate,
    mod2 = mod2$Estimate,
    mod3 = mod3$Estimate,
    beta2 = mod3_res$coefficients["x2_res"]
  )
},
  .options = furrr_options(seed = TRUE)
) %>%
  transpose()
```

Now, we can calculate a `diff_lp` over all the models in one big matrix of results (`do.call(rbind, results$modX)`):

```
diff_lp(do.call(rbind, results$mod1)) %>%
  gt() %>%
  fmt_scientific(-1, decimals = 2)
```

```
diff_lp(do.call(rbind, results$mod2)) %>%
  gt() %>%
  fmt_scientific(-1, decimals = 2)
```

```r
diff_lp(do.call(rbind, results$mod3)) %>%
  gt() %>%
  fmt_scientific(-1, decimals = 2)
```

For the beta, we can see, in average, how much it strays away from $\beta_2$.

```r
mean(abs(list_c(results$beta2) - beta2))
```