MATH-517: Assignment 3

Philipp Mayer

2025-09-03

Theoretical exercise

Define the quadratic form

$$\mathcal{Q}(\beta) = \sum_{i=1}^{n} K\left(\frac{X_i - x}{h}\right)(Y_i - \beta_0 - \beta_1(X_i - x))^2 = (Y - X\beta)^{\mathrm{T}} W (Y - X\beta),$$

where $W = \mathrm{Diag}\left(W_1, ..., W_n\right) = \mathrm{Diag}\left(K\left(\frac{X_1 - x}{h}\right), ..., K\left(\frac{X_n - x}{h}\right)\right)$.

Using matrix calculus, we have that $\frac{\partial \mathcal{Q}(\beta)}{\partial \beta} = 2X^{\mathrm{T}}WX\beta - 2X^{\mathrm{T}}WY$ so that the normal equations are given by $X^{\mathrm{T}}WX\hat{\beta} = X^{\mathrm{T}}WY$. Assuming that $X^{\mathrm{T}}WX$ is invertible, we immediately obtain from the normal equations that $\hat{\beta} = \left(X^{\mathrm{T}}WX\right)^{-1} X^{\mathrm{T}}WY$.

We will start off by compute some terms, $X^{\mathrm{T}}WX$ and $X^{\mathrm{T}}WY$ separately.

$$X^{\mathrm{T}}WX = \begin{bmatrix} 1 & \cdots & 1 \\ X_1 - x & \cdots & X_n - x \end{bmatrix} \begin{bmatrix} W_1 & & \\ & \ddots & \\ & & W_n \end{bmatrix} \begin{bmatrix} 1 & X_1 - x \\ \vdots & \vdots \\ 1 & X_n - x \end{bmatrix}$$

$$= \begin{bmatrix} 1 & \cdots & 1 \\ X_1 - x & \cdots & X_n - x \end{bmatrix} \begin{bmatrix} W_1 & W_1(X_1 - x) \\ \vdots & \vdots \\ W_n & W_n(X_n - x) \end{bmatrix}$$

such that

$$X^{\mathrm{T}}WX = \begin{bmatrix} \sum_{i=1}^{n} W_i & \sum_{i=1}^{n} W_i(X_i - x) \\ \sum_{i=1}^{n} W_i(X_i - x) & \sum_{i=1}^{n} W_i(X_i - x)^2 \end{bmatrix}$$

Define $\tilde{S}_{n,k}(x) := nhS_{n,k}(x)$ for $k = 0, 1, 2$, where $S_{n,k}(x)$ are defined as in the README. In view of this definition, $X^{\mathrm{T}}WX = \begin{bmatrix} \tilde{S}_{n,0} & \tilde{S}_{n,1} \\ \tilde{S}_{n,1} & \tilde{S}_{n,2} \end{bmatrix} = nh \begin{bmatrix} S_{n,0} & S_{n,1} \\ S_{n,1} & S_{n,2} \end{bmatrix}$.

We now turn to $X^{\mathrm{T}}WY$.

$$X^{\mathrm{T}}WY = \begin{bmatrix} 1 & \cdots & 1 \\ X_1 - x & \cdots & X_n - x \end{bmatrix} \begin{bmatrix} W_1 & & \\ & \ddots & \\ & & W_n \end{bmatrix} \begin{bmatrix} Y_1 \\ \vdots \\ Y_n \end{bmatrix} = \begin{bmatrix} W_1 & \cdots W_n & \\ W_1(X_1 - x) & \cdots & W_n(X_n - x) \end{bmatrix} \begin{bmatrix} Y_1 \\ \vdots \\ Y_n \end{bmatrix}$$

$$= \begin{bmatrix} \sum_{i=1}^n W_i Y_i \\ \sum_{i=1}^n W_i (X_i - x) Y_i \end{bmatrix}$$

Defining $T_0 = \sum_{i=1}^n W_i Y_i$ and $T_1 = \sum_{i=1}^n W_i (X_i - x) Y_i$, we have that $X^{\mathrm{T}}WY = \begin{bmatrix} T_0 & T_1 \end{bmatrix}^{\mathrm{T}}$.

We now invert $X^{\mathrm{T}}WX$. Since $X^{\mathrm{T}}WX$ is a 2×2 matrix, the inverse is

$$\begin{bmatrix} \tilde{S}_{n,0} & \tilde{S}_{n,1} \\ \tilde{S}_{n,1} & \tilde{S}_{n,2} \end{bmatrix}^{-1} = \frac{1}{D} \begin{bmatrix} \tilde{S}_{n,2} & -\tilde{S}_{n,1} \\ -\tilde{S}_{n,1} & \tilde{S}_{n,2} \end{bmatrix}, \text{ where } \; D = \tilde{S}_{n,0}\tilde{S}_{n,2} - \tilde{S}_{n,1}^2 \; \text{ is the determinent.}$$

Recall that we are interested in $\hat{\beta}_0$ and in view of the above reasoning,

$$\hat{\beta} = \frac{1}{D} \begin{bmatrix} \tilde{S}_{n,2} & -\tilde{S}_{n,1} \\ -\tilde{S}_{n,1} & \tilde{S}_{n,0} \end{bmatrix} \begin{bmatrix} T_0 \\ T_1 \end{bmatrix} \; \text{ such that } \; \hat{\beta}_0 = \frac{1}{D} \left(\tilde{S}_{n,2} T_0 - \tilde{S}_{n,1} T_1 \right) = \sum_{i=1}^n \frac{W_i \left(\tilde{S}_{n,2} - \tilde{S}_{n,1}(X_i - x) \right)}{D} Y_i$$

Recall that $\tilde{S}_{n,k} = nhS_{n,k}$. It follows that $D = (nh)^2 \left(S_{n,0} S_{n,2} - S_{n,1}^2 \right)$. It immediately follows that

$$\hat{\beta}_0 = \sum_{i=1}^n \frac{nh}{(nh)^2} \frac{W_i \left(S_{n,2} - S_{n,1}(X_i - x) \right)}{S_{n,0} S_{n,2} - S_{n,1}^2} Y_i = \sum_{i=1}^n \frac{1}{nh} \frac{W_i (S_{n,2} - S_{n,1}(X_i - x)}{S_{n,0} S_{n,2} - S_{n,1}^2} Y_i = \sum_{i=1}^n w_{n,i} Y_i,$$

where

$$w_{n,i} = w_{n,i}(x) = \frac{1}{nh} \frac{K\left(\frac{X_i - x}{h}\right) \left(S_{n,2}(x) - S_{n,1}(x)(X_i - x) \right)}{S_{n,0}(x) S_{n,2}(x) - S_{n,1}(x)^2}.$$

With this result, we answer both question (1) and (2) at the same time.

It remains to show that $\sum_{i=1}^{n} w_{n,i}(x) = 1$. We proceed as follows:

$$\sum_{i=1}^{n} w_{n,i}(x) = \frac{1}{nh} \sum_{i=1}^{n} \frac{W_i \left(S_{n,2}(x) - (X_i - x) S_{n,1}(x) \right)}{S_{n,0}(x) S_{n,2}(x) - S_{n,1}(x)^2}$$

$$= \frac{1}{nh} \frac{S_{n,2}(x)}{S_{n,0}(x) S_{n,2}(x) - S_{n,1}(x)^2} \sum_{i=1}^{n} W_i - \frac{1}{nh} \frac{S_{n,1}(x)}{S_{n,0}(x) S_{n,2}(x) - S_{n,1}(x)^2} \sum_{i=1}^{n} W_i (X_i - x)$$

In view of the definitions of $S_{n,0}(x)$ and $S_{n,1}(x)$, we obtain

$$\sum_{i=1}^{n} w_{n,i}(x) = \frac{S_{n,2}(x) S_{n,0}(x)}{S_{n,0}(x) S_{n,2}(x) - S_{n,1}(x)^2} - \frac{S_{n,1}(x)^2}{S_{n,0}(x) S_{n,2}(x) - S_{n,1}(x)^2} = 1,$$

which concludes the third (3) part of the exercise.

This concludes the theoretical exercise.

Practical exercise

Unless stated otherwise, your answer to the practical part should include the following elements:

- Description of the aim of the simulation study.
- Description of the different quantities that intervene in your simulation study. Explain how these quantities (fixed or random) are defined and the reasoning behind the choices you made.
- Description of your findings using appropriate graphics and/or tables (with a caption!) that are well commented in the text.

The code should not appear in the PDF report, unless there is a specific reason to include it.

Structure and details:

Aim of the simulation study:

The goal is to observe the impact of various parameters on the estimation of the IMSE-optimal bandwith h_{IMSE}, σ^2 and θ_{22}. In particular how it varies with respect to

- various values (combinations) of the Beta function, $\alpha, \beta > 0$ (see below)
- various values (combinations) of sample size n and block size / number of blocks N (see below)

Fixed quantities:

- $\alpha, \beta > 0$ are the parameters of the Beta distribution, they're are usually named `alpha` and `beta` in the code. We have a sample size n and a number of blocks N, they're usually denoted descriptively, for example `sample_size` or `size` and `number_of_blocks`. Note that in the README, we consider the size of the blocks instead of the number [??????????????????? what do i do s??????????????????????????]
- ADD IF NECESSARY

Code architecture:

The technical functions are located in a file `functions.py` with usual alias `fct`. There, we have the following functions:

- `generate_sample` takes inputs `alpha, beta, n_samples, sigma_2` which respectively represent α, β, the number of samples n and the variance of the error ϵ, σ^2. This function returns two one-dimensional arrays, `covariate` and `response` representing X and Y respectively.
- `estimate_parameters` takes inputs `covariate, response, bandwith, p, reg`. They represent respectively X, Y, a default bandwith needed to perform an initial KDE and p which represents the order. Note that `p`$= 1$ by default and `reg` is set to `1e-5`. It computes the estimated $\hat{\beta}$ by solving the normal equation. To efficiently compute the solution, we use Cholesky decomposition and `np.linalg.solve` instead of `np.linalg.inv`, which is significantly more stable and efficient in this situation. For even more stability, we add a small regularization (`reg`) before using `np.linalg.solve`. Throughout the calls of the functions, the `reg` parameter is never changed and remains set to its default value of `1e-5`. The returns is a function `beta_est` that, given a array of covariates, computes the fitted values.
- `estimate_sigma_theta` takes inputs `covariate, repsonse, bandwith, N_blocks` which respectively represent X, Y, a default bandwith needed to perform the initial KDE and the number of blocks to compute estimates of $\hat{\theta}_{22}$ and $\hat{\sigma}^2$. Note that in the README, it is usually mentioned the size of the blocks instead of the number. We decided to implement in terms of the number of blocks. This choice is due to (my) reasoning logic. The computation of the size, given the number of blocks is given by `size = sample_size / N_blocks`. To compute blocks, we compute the size, and then split the data in blocks `[B1 | ... | Bk | Bextra]` `Bextra` is the remaining number of samples. This function returns an estimate of θ_{22} and σ^2, following the procedure recommended in the README.
- `est_h_IMSE_support` takes inputs `sigma_2_hat, theta_22, size, covariate` as inputs which respectively represent $\hat{\sigma}^2$, $\hat{\theta}_{22}$ and the sample size n. It returns h_{IMSE} as given in the README. Note that `size` is not a necessary argument as `size=len(covariate)`. This choice was made for clarity. The `covariate` array is a necessary argument as the support of the realizations is given by `support=max(covariate)-min(covariate)` (note

that this is always a nonnegative quantity and a positive quantity with probability one). This function returns an estimate of h_{IMSE}.

- `compute_mallow_C_p` takes inputs `covariate, response, fixed_bandwith, N_blocks` (as previously seen). It computes $C_p(N)$ as suggested in the README and returns its value.

- `optimal_mallow` takes inputs `covariate, repsonse, fixed_bandwith, max_N_blocks`. The first two inputs are as seen previously and the third is the same as `bandwith` seen previously. The last argument represents the maximal number of blocks on which the function computes the C_p statistics (for m "model" selection). The function returns the number of blocks that minimized the Mallow C_p statistic. The function calls `compute_mallow_C_p` for each block size.

- `h_IMSE_Cp_optimized` takes inputs `covariate, reponse, number_of_samples, default_bandwith, max_number_of_blocks`, which respectively represent X, Y, n, the fixed bandwith as before (previously called `bandwith` or `fixed_bandwith` and the maximum number of blocks to test (previously called `max_N_blocks`. The function calls `optimal_mallow` and re-estimates σ^2 and θ_{22}, given the Mallow-optimal number of blocks (the return of `optimal_mallow`). This function returns an estimation of σ^2 and θ_{22} with Mallow-optimal selection of the number of blocks.

- `simulate` takes inputs `alpha, beta, number_of_samples, error_variance, default_bandwith, number_of_blocks`, which respectively represent $\alpha, \beta, n, \sigma^2$, a default bandwith (as previously seen) and a number of blocks. It first calls `generate_sample` with $\alpha, \beta, n, \sigma^2$, which returns `covariate, response`. With that return, it calls `estimate_sigma_theta` with the default bandwith and the number of blocks. FInally, it calls `est_h_IMSE_support` with the computed quantities. This function returns the estimations of h_{IMSE}, σ^2 and θ_{22}. We mostly use this function to ease the production of plots.

The plotting functions are accessible through a file called `plot_results.py`. There, one can use any of the functions below:

- `plot_simple_fit` takes inputs `covariate, response, fixed_bandwith, sigma_2`, representing X, Y, the fixed bandwith as previously seen and σ^2 (for the plot title). It returns a plot showing visually how the choice of bandwith impacts the fits. To produce this plot simply, refer to the subsection below "**How to simply plot results**". An example of call would be:

  - `py plot_results.py simple_plot 0.1 0.2 10000 0.2 1`

- `plot_alpha_beta_impact` takes inputs `alphas, betas, error_variance, default_bandwith, number_of_blocks`, which respectively represent a list of alphas $[\alpha_1, ..., \alpha_{n_\alpha}]$, as list of betas $[\beta_1, ..., \beta_{n_\beta}]$, the default bandwith as seen previously and `error_variance` represents σ^2. This function produces three heat maps of the values of estimated

h_{IMSE}, σ^2 and θ_{22} for different values of α_i, β_j. To produce this plot simply, refer to the subsection below "**How to simply plot results**". An example of call would be:

   - `py plot_results.py alpha_beta_impact 100 1.0 0.1 5`

- `plot_sample_blocks_impact` takes inputs `number_of_samples_range, number_of_blocks, alpha, beta, error_variance, default_bandwith`, which respectively represent a list $[n_1, ..., n_{k_n}]$ of values for the number of samples n, a list $[N_1, ..., N_{k_N}]$ of values for the number of blocks N, fixed values of α, β and σ^2. This function produces three heat maps of the values of estimated h_{IMSE}, σ^2 and θ_{22} for different values of n_i and N_j. To produce this plot simply, refer to the subsection below "**How to simply plot results**". An example of call would be:

   - `py plot_results.py sample_blocks_impact 0.1 0.2 1.0 0.1`

- `plot_sample_size_impact_mallow` takes inputs `alpha, beta, max_number_of_samples, step, error_variance, default_bandwith, max_number_of_blocks`, as seen previously. `max_number_of_blocks` is set to `10` by default. The additional new parameter represents the number of steps between two different sample sizes. The list of sample sizes to test are then given by `n_range = np.arange(start=default_start, stop=max_number_of_samples, step=step)`. This function produces three (sub) plots representing how the estimations of h_{IMSE}, σ^2 and θ_{22} evolve when computed on n samples with Mallow-optimal number of blocks. To produce this plot simply, refer to the subsection below "**How to simply plot results**". An example of call would be:

   - `py plot_results.py sample_size_mallow_evolution 0.1 0.2 2000 10 1.0 0.1`

How to simply plot results:

To call them, use the console with one of the following keywords `key`$\in$ { `simple_plot, alpha_beta_impact, sample_blocks_impact, plot_sample_size_impact_mallow` } and call as follows: `py plot_results.py key parameter1 parameter2 parameter3 ...`

where `parameter1 parameter2 ...` are the functions parameters, detailed below:

- when `key=simple_plot`: The parameters are in the following order `alpha, beta, number_of_samples, fixed_bandwith, variance`, representing respectively α, β, n, the fixed bandwith (as previously mentioned) and σ^2.
- when `key=alpha_beta_impact`: The parameters are in the following order `number_of_samples, error_variance, default_bandwith, number_of_blocks`, representing respectively the number of samples n to test, σ^2, the default bandwith (as previously mentioned) and the number of blocks.
- when `key=sample_blocks_impact`: The parameters are in the following order `alpha, beta, error_variance, default_bandwith`, as previously seen.

- when `key=plot_sample_size_impact_mallow`: The parameters are in the following order `alpha, beta, max_number_of_samples, step, error_variance, default_bandwith`.

When a call is successful, the console will display `TASK:<followed by the actual task it is performing>` and will print `TASK completed` when the plot has been produced.

Findings and reports:

We start off with a simple visualization of the setup for fixed α, β, n. The plot below showcases how different bandwith choices visually impact the fit. In orange is the true $m(x)$. In red is a (relatively good fitting) line. We observe that as the bandwith goes closer to zero, the fits become better. However, when to close to zero, the fitted function appears to be to sensible and does not provide a robust fit to the truth.

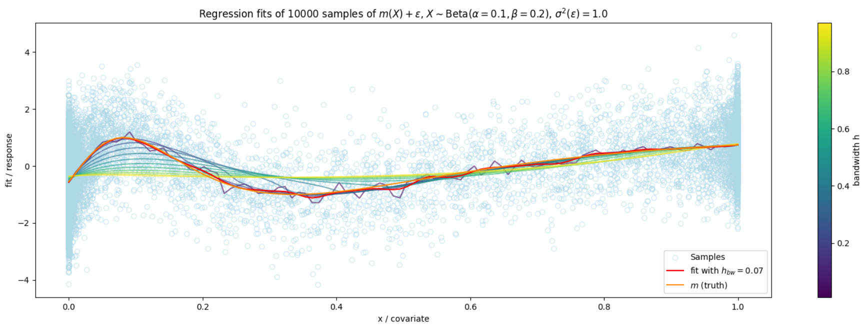

Figure 1: Visual plot of bandwith impact, call of `py plot_results.py simple_plot 0.1 0.2 10000 0.2 1`

This guides us with the belief that an optimal bandwith may have to be chosen close to zero, but not too close either (at least for these parameter choices).

Here below are heatmaps of the results of estimations of $h_{IMSE}, \sigma^2, \theta_{22}$ against various values of α, β. The x-axis represents β and the y-axis represents α. Here the alphas and betas are chosen in `np.arange(0.1, 10, step = 0.5)`. We observe what seems to be a boundary effect when either α or β is close to zero on the estimation of h_{IMSE} and θ_{22} in particular. For h_{IMSE}, when α or β (or both) are close to zero, the values of h_{IMSE} are of larger magnitude in comparison to the other cases. The inverted observation can be made for θ_{22}'s estimation where the values are lower when α, β are close to zero in comparison to the other case.

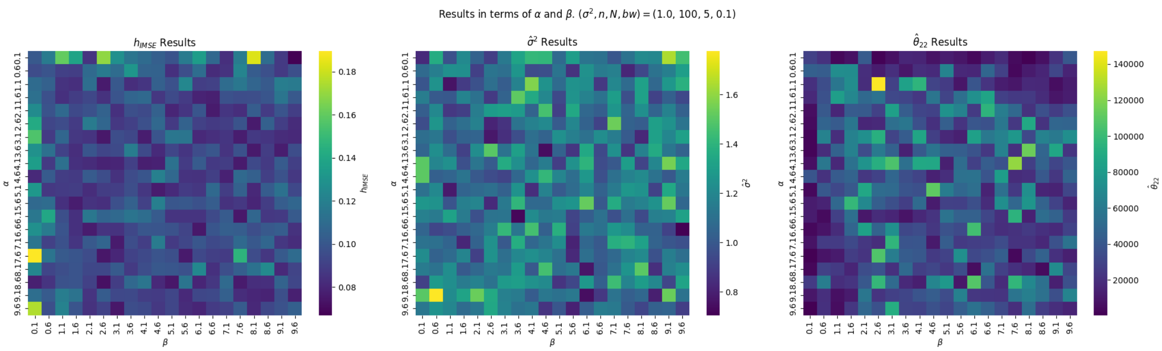

Figure 2: Impact of the choices of alpha and beta on estimations, call of `py plot_results.py alpha_beta_impact 100 1.0 0.1 5`

To better emphasize on this boundary effect, we recomputed the heatmaps for α, β taking values in `np.arange(0.01, 5, step = 0.5)`. There, the boundary effect on the estimation of h_{IMSE} and θ_{22} are even more noticeable.

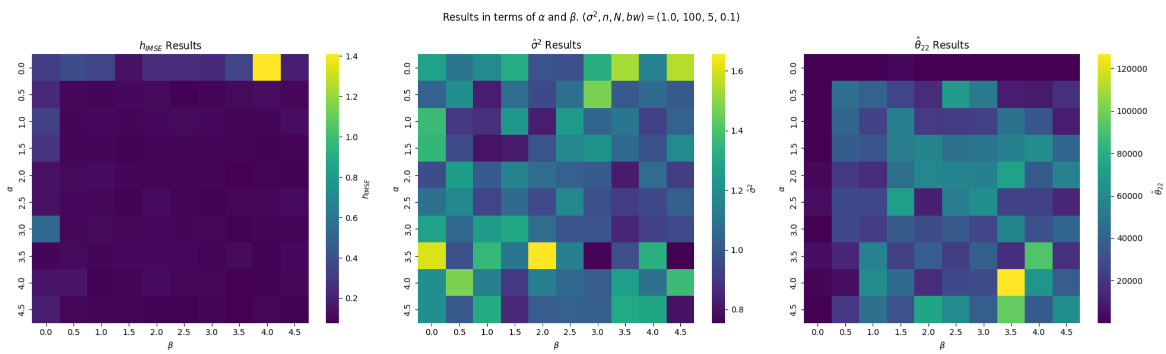

Figure 3: Impact of the choices of alpha and betas, close to zero

Note that in both cases, the estimations of σ^2 remain relatively stable between 0.8 and 1.2 approximately.

To complete the observations around the impacts of α, β on the estimations, we perform a last run for α, β taking values in `np.arange(0.01, 1, step = 0.01)`. There, we observe a fade from the left and upper boundaries to the center for θ_{22}, and a strong boundary effect on h_{IMSE}, as already expected. We additionally observe that the estimations of h_{IMSE} remain very stable for values of α, β far enough from zero (both at the same time) and that the estimations of σ^2 are reasonably stable on average.

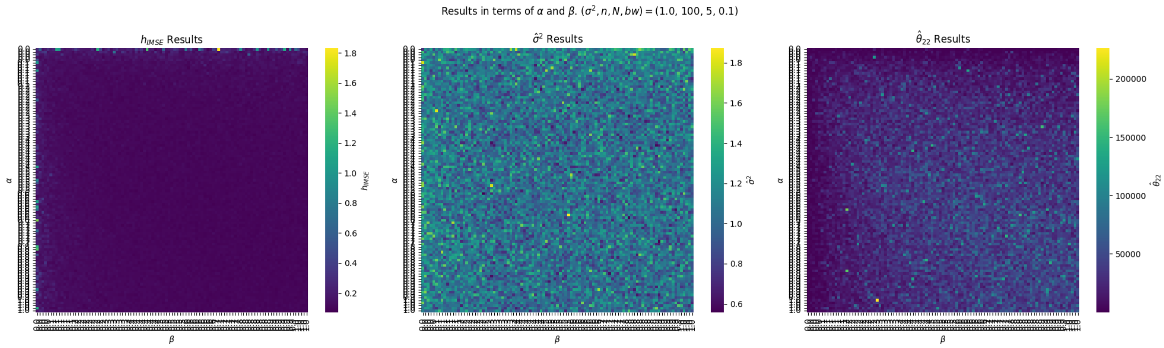

Figure 4: Impact of the choices of alpha and beta with increased granularity and closer to zero

In conclusion,

- The estimation of h_{IMSE}, θ_{22} are more stable for values of α, β far enough from zero (boundary effect). The estimation of h_{IMSE} has very few extreme values (so low variance) for values of α, β large enough.
- The estimation of σ^2 appears to not be affected a lot by changes of α, β.
- initial estimations are that h_{IMSE} is around 0.2 on average, σ^2 is, as expected, around 1.0 and θ_{22} appears to be around 50000 on average.

NB: Note that to produce these variations, one must directly comment / uncomment parts of the code as well as change export name of the `.png` files. The default computation is made for values of α, β in `np.arange(0.1, 10, step = 0.5)`.

We now take a look at how the number of blocks (or similarly the size of the blocks) and the sample size impact the estimations.

As already mentioned, we think of number of blocks instead of size of blocks. As the number of blocks decrease, the size increases and vice-versa. First notice that we obtain on-average inconsistent estimations of h_{IMSE} and σ^2 when the sample size is low and especially when n s small and N is large. For θ_{22} on the other hand, the computations appear to be inconsistent in comparison to the average when the number of blocks is small. It becomes more stable as the number of blocks increase. This is an observation that one can also make for h_{IMSE} and σ^2.

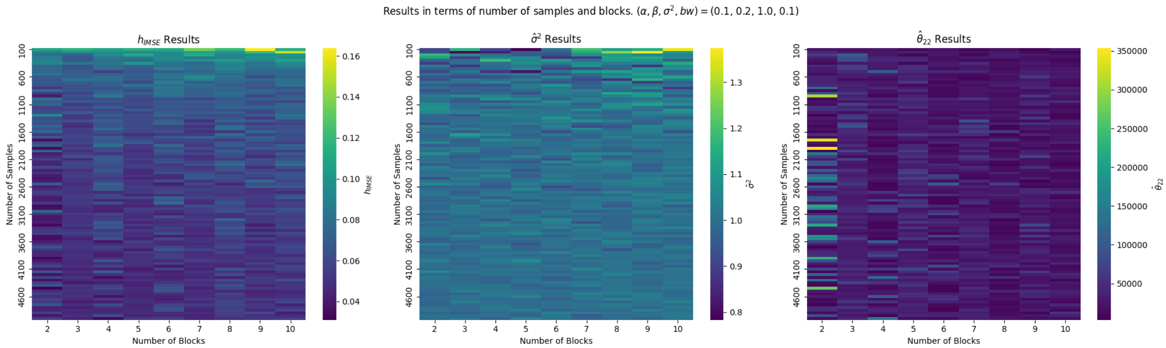

Figure 5: Impact of sample size and number of blocks on estimations (color map)

In conclusion,

- As expected, the sample size positively increases the on-average consistency of the estimations.
- The size of the blocks decreases (and the number of blocks increases) the estimations appear to become stable when n (the sample size) is large enough.

Let us now take a look at how the estimations behave against the sample size when the number of blocks (or similarly the size) is selected using Mallow's C_p selection criterion. This is a reasonable way to compare the impact of sample sizes alone.

We observe, as expected and previously observed, that small sample sized (even with optimal number of blocks) yield unreasonable estimations in comparison to the average. Moreover, the estimation of h_{IMSE} appears to average around the value of 0.07 after the elbow. The estimation of σ^2 also becomes stable and converges to, it appears, 1.0 (as expected). Furthermore, as already observed previously, the estimation of θ_{22} is relatively unstable.

Figure 6: Estimations against sample size with Mallow-selected block number / size

Here below is the call of `py plot_results.py simple_plot 0.1 0.2 10000 0.07 1` with the bandwith of 0.7 mentioned below.

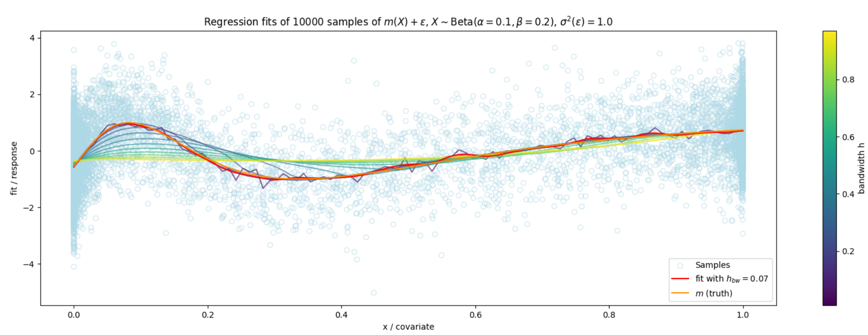

Figure 7: Plot with bandwith of 0.7

NB: Note that to produce these variations, one must directly change export name of the ***.png*** *files.*

<u>Answers to suggested questions to address:</u>

1. How does h_{IMSE} evolve as N grows and why could this be:
2. Should N depend on n and why:
3. What happens when the number of observations varies a lot between different regions in the support of X and link to Beta(α, β):