Problem Set: Non Parametric Statistics: Group 18

Cesar Conejo Villalobos, Xavier Bryant

3/30/2021

Question 1

Question 2

Compare the MISE and AMISE criteria in three densities in `nor1mix` of your choice.

1. Code (2.33) and the AMISE expression for the normal kernel, and compare the two error curves.

2. Compare them for $n = 100, 200, 500$, adding a vertical line to represent the h_{MISE} and h_{AMISE} bandwidths. Describe in detail the results and the major takeaways.

For this exercise, we require some mathematical ideas that we will develop briefly.

We start with the KDE estimator $\hat{f}(x; h) = \sum_{i=1}^{n} K_h(x - X_i)$. The expectation and variance for this estimator are given by the following expressions:

(1) $E[\hat{f}(x; h)] = (K_h * f)(x)$.

(2) $\mathrm{Var}[\hat{f}(x; h)] = \frac{1}{n}((K_h^2 * f)(x) - (K_h * f)^2(x))$

Then, we develop some assymptotic expressions for (1) and (2):

(3) $E[\hat{f}(x; h)] - f(x) = \mathrm{Bias}[\hat{f}(x; h)] = \frac{1}{2}\mu_2(K)f''(x)h^2 + o(h^2)$

(4) $\mathrm{Var}[\hat{f}(x; h)] = \frac{R(K)}{nh}f(x) + o((nh)^{-1})$

Then, from whe equations (3) and (4) we obtain the following expression for the MSE:

(5) $\mathrm{MSE}[\hat{f}(x; h)] = \frac{\mu_2^2(K)}{4}(f''(x))^2 h^4 + \frac{R(K)}{nh}f(x) + o(h^4 + (nh)^{-1})$

In equations (3), (4) and (5) we have:

(6) Second order moment of K: $\mu_2(K) := \int z^2 K(z)dz$

(7) Squared integral of kernel: $R(K) := \int (K(x))^2 dx$

Then, we use $\mathrm{MISE}[\hat{f}(\cdot; h)]$ as global error criteria for measuring the performance of $\hat{f}$ in relation to the target density f.

(8) $\mathrm{MISE}[\hat{f}(\cdot; h)] = \int \mathrm{MSE}[\hat{f}(x; h)]$

Therefore, we obtain the following asymptotic expansion for the MISE:

(9) $\mathrm{MISE}[\hat{f}(\cdot; h)] = \frac{1}{4}\mu_2^2(K)R(f'')h^4 + \frac{R(K)}{nh} + o(h^4 + (nh)^{-1})$

We define the dominating part of equation (9) as $\mathrm{AMISE}[\hat{f}(\cdot; h)]$. In particular:

(10) $\mathrm{AMISE}[\hat{f}(\cdot; h)] = \frac{1}{4}\mu_2^2(K)R(f'')h^4 + \frac{R(K)}{nh}$

with the expression $R(f'')$ given by:

(11) $R(f'') = \int (f''(x))^2 dx$

Finally, the bandwidth the minimizes the AMISE is:

(12) $h_{\mathrm{AMISE}} = \left[\frac{R(K)}{\mu_2^2(K)R(f'')n} \right]^{1/5}$

Now, we consider our particular case of study. In this case, we *reduce* our analysis considering:

a) A normal kernel $K_h(\cdot)$ with distribution $\mathcal{N}(0, 1)$

b) The density function f is based on the family of normal r-mixtures:

(13) $f(x; \mu, \sigma, \boldsymbol{w}) = \sum_{j=1}^{r} w_j \phi_{\sigma_j}(x - \mu_j)$

where $w_j \geq 0$, $j = 1, ..., r$ and $\sum_{j=1}^{r} w_j = 1$.

With these two expressions, we can obtain a specific value for the AMISE in equation (10). With assumption a), we obtain the following expressions for equations (6) and (7)

(6.1) Second order moment of K: $\mu_2(K) = 1$

(7.1) Squared integral of kernel: $R(K) = \frac{1}{2\sqrt{\pi}}$

Expression for equation (11) can be obtained from the following adaption of expression given in *Theorem 4.1* given by *Marron and Wand (1992)* (In particular, taking $s = 2$ in the expression given in the paper)

(11.1) $R(f'') = \int (f''(x))^2 dx = \sum_{j=1}^{r} \sum_{j'=1}^{r} w_j w_{j'} \phi_{\sigma_{jj'}}^4 (\mu_j - \mu_{j'})$

where $\sigma_{jj'} = \sigma_j^2 + \sigma_{j'}^{2}{}^{1/2}$. Moreover, on order to tackle this expression, we use the *probabilist's Hermite polynomial* of order 4: $\phi^{(4)} = \phi(x)H_4(x) = \phi(x) * (x^4 - 6x^2 + 3)$.

With equations (6.1), (7.1) and (11.1), we obtain the reduced form of the AMISE:

(10.1) $\text{AMISE}[\hat{f}(\cdot; h)] = \frac{1}{4} R(f'')h^4 + \frac{1}{2nh\sqrt{\pi}}$

With optimal bandwidth h_{AMISE} given by:

(12.1) $h_{\text{AMISE}} = \left[\frac{(2\sqrt{\pi})^{-1}}{R(f'')n} \right]^{1/5}$

Additionally, under these assumptions, we obtain an explicit and exact MISE expression of equation (8):

(14) $\text{MISE}_r[\hat{f}(\cdot; h)] = (2\sqrt{\pi}nh)^{-1} + \boldsymbol{w}'\{(1 - n^{-1})\Omega_2 - \Omega_1 + \Omega_0\}\boldsymbol{w}$

Where:

(15) $(\Omega_a)_{i,j} = \phi_{(ah^2 + \sigma_i^2 + \sigma_j^2)^{1/2}}(\mu_i - \mu_j)$ for $i, j = 1, ..., r$

Finally, we can proceed with a numeric approach over equation (14) and obtain:

(16) $\arg\min_{h>0}\text{MISE}[\hat{f}(\cdot; h)]$

With this mathematical review, we are able to compare the MISE and AMISE criteria. We code the following functions isn `R`:

- `omega_a()`: Compute the matrix $(\Omega_a)_{i,j}$ defined in equation (15).

- `omega()`: Compute the **scalar** given by the following vector and matrix operations of equation (14): $\Omega = \boldsymbol{w}'\{(1 - n^{-1})\Omega_2 - \Omega_1 + \Omega_0\}\boldsymbol{w}$

- `MISE()`: Useful for computing the expression of equation (14).

- `AMISE()`: Useful for computing the expression of equation (10.1). We also code the auxiliary functions `Hermite_4()` and `R_f2()` related to equation (11.1).

- Value of h_{AMISE} is obtained from the code function `h_AMISE_n()`. Value of h_{MISE} is obtained numerically from equations (15) and (15) using `optimize()`.

We consider in particular the following three densities of `nor1mix` whose parameter are available in R Help

- `MW.nm1`: Gaussian $\mathcal{N}(0, 1)$

- `MW.nm2`: Skewed $.2\mathcal{N}(-.3, 1.44) + .2\mathcal{N}(.3, .64) + .6\mathcal{N}(1, 4/9)$

- `MW.nm6`: Bimodal $.5\mathcal{N}(-1, 4/9) + .5\mathcal{N}(1, 4/9)$

In the case using the dataset `nor1mix::MW.nm1` figure 1 shows the histograms for the sample distribution for $n = 100, 200, 500$. As expected, behavior of the sample improve when we increase the number of observations.
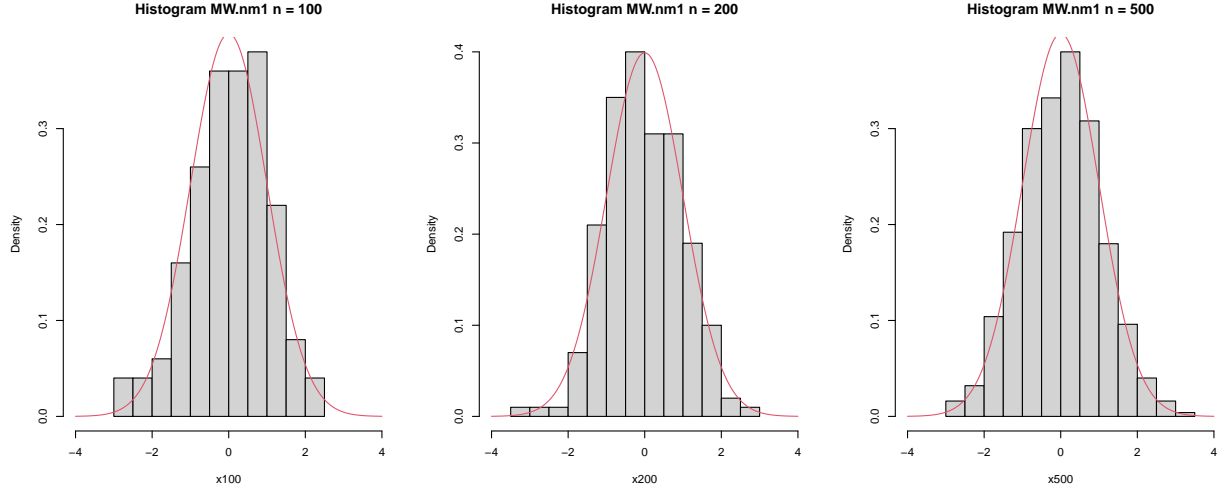
Figure 1: Histograms from sample obtained from n = 100, 200 and 500 for object MW.nm1

Figure 2 shows the MISE and AMISE related to `nor1mix::MW.nm1`. The values of h_{AMISE} are 0.422, 0.367 and `round(h_AMISE_500,3)`. On the hand values of h_{MISE} are 0.445, 0.383 and `round(h_MISE_500$minimum,3))` for $n = 100, 200, 500$. In the three curves, when we reduce the number of the bandwidth, the MISE and AMISE values are quite similar. Additionally, we can see the effect of increasing the number of samples n in the y axis of the plot. (The scaled is reduced). So, the difference between both

In this case, when we increase the number of samples n and reduce the value of the bandwidth, the curves for MISE and
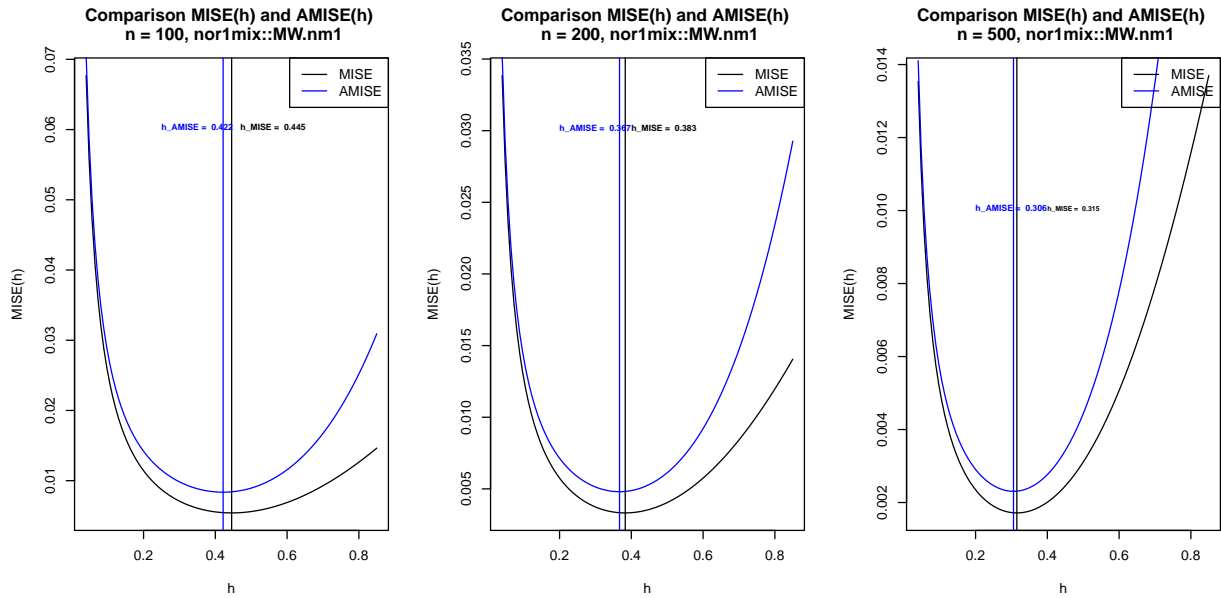

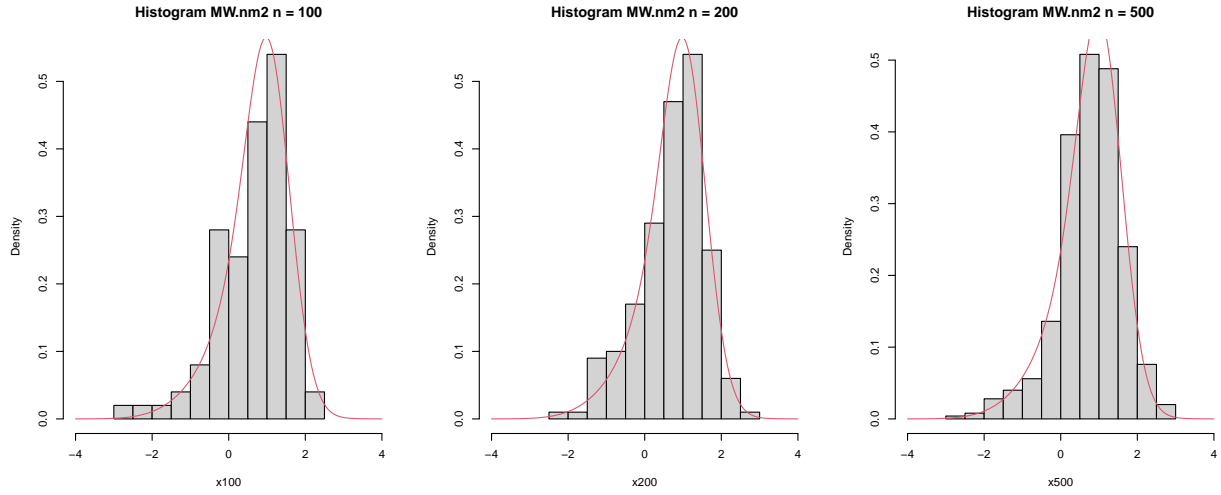Figure 2: MISE and AMISE for range bandwith between 0.04 and 0.85

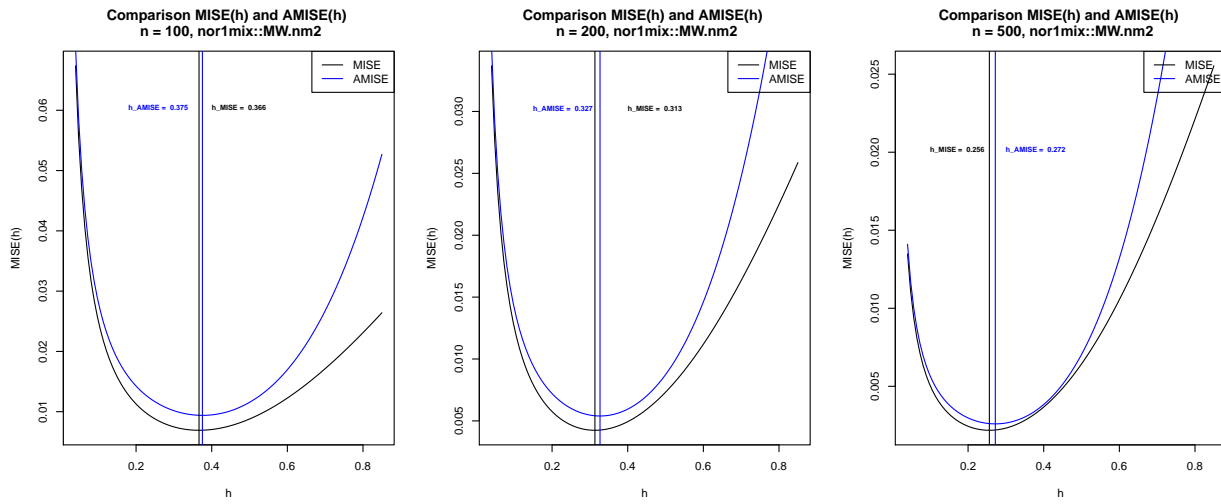Figure 3: Histograms from sample obtained from n = 100, 200 and 500 for object MW.nm2 (Skewed)

Figure 4: MISE and AMISE for range bandwith between 0.04 and 0.85

Question 3

Adapt the `np_pred_CI` function to include the argument `type_boot`, which can take either the value `"naive"` or `"wild"`.

1) If `type_boot = "wild"`, then the function must perform the wild bootstrap algorithm, implemented from scratch following substeps i–iv.

2) Compare and validate the correct behavior of the confidence intervals, for the two specifications of type_boot, in the model considered in Exercise 5.8 (without performing the full simulation study).

For this question we describe briefly the necessary steps related to the implementation of the wild-bootstrap:

First of all, for a given sample with n observations, the predicted (fitted) values are given by the expression: $\hat{Y}_i := \hat{m}(X_i; q, h)$ with $i = 1, ..., n$.

Then, the uncertainty of $\hat{m}(x, q, h)$ (this includes the consideration of new set of samples) can be measure based on two approaches:

1. Asymptotic approach: $\hat{m}(x; q, h) \pm \hat{se}(\hat{m}(x; q, h))$ where $\hat{se}(\hat{m}(x; q, h))$ is the asymptotic estimation of the standard deviation of $\hat{m}(x; q, h)$

2. Bootstrap: In this case, we consider two types of bootstrap:

- Naive Bootstrap: From the original sample, we create a new set of IID observations called the *Bootstrap sample*

- Wild Bootstrap: This technique is focused on resampling the residuals. Similar to the *residual bootstrap* in which we fixed the covariate X_i and generate a new value of Y_i using the fitted model and the *noise* from sampling the residuals, additional we generate IID random variables V_i (random variable with mean zero and variance 1) and then we can perturb the residuals. This modification improves the stability of the computations, especially in the presence of heteroskedasticity.

The algorithm of the computation of the wild boostrap is given in the following steps:

1. Compute $\hat{m}(x; q, h) = \sum_{i=1}^{n} W_i^q(x) Y_i$ from the original sample $(X_1, Y_1), ..., (X_n, Y_n)$

2. Enter the wild bootstrap. For $b = 1, ..., B$.

   i. Simulate $V_1^{*b}, ..., V_n^{*b}$ to be iid copies of V such that $E[V] = 0$ and $Var[V] = 1$,

   ii. Compute the *pertubed residuals* $e_i^{*b} = \hat{e}_i V_i^{*b}$ where $Y_i^{*b} := \hat{m}(X_i; q, h) + e_i^{*b}$ for $i = 1, ..., n$.

   iii. Obtain the bootstrap sample: $(X_1, Y_1^{*b}), ..., (X_n, Y_n^{*b})$ where $Y_i^{*b} := \hat{m}(X_i; q, h) + e_i^{*b}$ with $i = 1, ..., n$.

   iv. Compute $\hat{m}^{*b}(x; q, h) = \sum_{i=1}^{n} W_i^q(x) Y_i^{*b}$ from $(X_1^*, Y_1^{*b}), ..., (X_n^8, Y_n^{*b})$

As a result, we modified the function `np_pred_CI`. The inputs of the original function are the following:

1. `npfit`: A np::npreg object (npfit)

2. `exdat`: Values of the predictors were to carry out prediction (exdat)

3. `B`: Number of bootstrap iterations.

4.`conf`: Range of confidence interval

5. `type_CI`: Type of confidence interval. (Normal standard or quantiles)

Additionally, we add three new inputs:

6. `type_boot`: Type of bootstrap procedure. Options: `Naive` and `Wild` bootstrap

7. `perturbed_res`: For the case of `Wild` Bootstrap, we can choose the type of perturbation. As explained before, any random variable with mean 0 and variance 1 can be used. As a result, we use two possible perturbation:

7.1. `normal`: Based on the normal distribution $V_i \sim \mathcal{N}(0,1)$

7.2 `golden`: Based on the *golden section binary variable* $P[V = 1 - \phi] = p$, $P[V = \phi] = 1 - p$ and $p = \frac{\phi+2}{5}$

  8. `seed`: Used for reproducibility. Default option is `seed = 42`.

The code of the new version of `np_pred_CI` is given below. We create a new `if` condition, in order to compute the type of bootstrap resampling method chosen by the user. In the case of the `wild` bootstrap, we focus on resampling the residuals obtained from the difference of the fitted value $\hat{Y}_i$ and the *real* values Y_i. Moreover, inside the option `type_boot == wild`, there is an `if` condition with the two possible alternatives for computing the perturbed residuals.

```
# Function to predict and compute confidence intervals for m(x).

# 1) Inputs

## 1.1) npfit: A np::npreg object (npfit)
## 1.2) exdat: Values of the predictors where to carry out prediction (exdat)
## 1.3) B:     Number of bootstrap iterations.
## 1.4) conf:  Range of confidence interval
## 1.5) type_CI: Type of confidence interval. (Based on normal standard or quantiles)
## 1.6) type_boot: Type of bootstrap procedure. Options Naive and Wild bootstrap
## 1.7) perturbed_res: Valid only for Wild Bootstrap. Type of perturbation on the residuals.
## Options are "normal"or "golden"
## 1.8) seed: Used for reproducibility. Default option is seed = 42.

# 2) Outputs

## 2.1) exdat: Values of the predictors where to carry out prediction
## 2.2) m_hat: Predicted regression
## 2.3) lwr:   Lower confidence interval
## 2.4) upr:   Upper confidence interval

np_pred_CI <- function(npfit,
                       exdat,
                       B = 200,
                       conf = 0.95,
                       type_CI = c("standard", "quantiles")[1],
                       type_boot = c("naive", "wild")[1],
                       perturbed_res = c("normal", "golden")[1],
                       seed = 42) {

  # Fix seed
  set.seed(seed)

  # Extract predictors
  xdat <- npfit$eval

  # Extract response, using a trick from np::npplot.rbandwidth
  tt <- terms(npfit$bws)
  tmf <- npfit$bws$call[c(1, match(c("formula", "data"),
                                   names(npfit$bws$call)))]
  tmf[[1]] <- as.name("model.frame")
  tmf[["formula"]] <- tt
  tmf <- eval(tmf, envir = environment(tt))
  ydat <- model.response(tmf)
```

```r
# Predictions m_hat from the original sample
m_hat <- np::npreg(txdat = xdat,
                   tydat = ydat,
                   exdat = exdat,
                   bws   = npfit$bws)$mean

if (type_boot == "naive") {

  # Function for performing naive bootstrap
  boot_function_naive <- function(data, indices) {
    np::npreg(txdat = xdat[indices,],
              tydat = ydat[indices],
              exdat = exdat,
              bws   = npfit$bws)$mean
  }

  # Carry out the bootstrap estimator
  m_hat_star <- boot::boot(data = data.frame(xdat),
                           statistic = boot_function_naive,
                           R = B)$t

} else if (type_boot == "wild") {

  # Sample size of the predictors
  n <- length(xdat)

  # Y fitted
  Y_hat <- npfit$mean

  # Ordinary residuals
  residuals_0 <- Y_hat - ydat

  # Type of perturbation
  if(perturbed_res == "normal"){

    # Function for performing wild bootstrap
    boot_function_wild <- function(data, indices) {

      # Step i: Simulate V_{i} copies of V (Mean 0 and variance 1)
      V_n <- rnorm(n)

      # Step iii. Obtain the bootstrap sample
      ydat_bt <- Y_hat + data[indices]*V_n

      np::npreg(txdat = xdat,
                tydat = ydat_bt,
                exdat = exdat,
                bws = npfit$bws)$mean
    }

    # Step iv. Carry out the wild bootstrap estimator
    m_hat_star <- boot::boot(data = residuals_0,
                             statistic = boot_function_wild,
```

```r
                          R = B)$t

  } else if(perturbed_res == "golden"){

    # Function for performing wild bootstrap
    boot_function_wild <- function(data, indices) {

      # Step i: Simulate V_{i} copies of V (Mean 0 and variance 1)
      phi <- (1 + sqrt(5))/2
      prob <- (phi + 2)/5

      golden <- sample(x = c(1-phi,phi), size = n, prob = c(prob, 1 - prob), replace=T)

      # Step iii. Obtain the bootstrap sample
      ydat_bt <- Y_hat + data[indices]*golden

      np::npreg(txdat = xdat,
                tydat = ydat_bt,
                exdat = exdat,
                bws = npfit$bws)$mean
    }

    # Step iv. Carry out the wild bootstrap estimator
    m_hat_star <- boot::boot(data = residuals_0,
                             statistic = boot_function_wild,
                             R = B)$t

  }

  else{stop("Incorrect type of peturbation")}

}else{stop("Incorrect type_boot")}

# Confidence intervals
alpha <- 1 - conf

if (type_CI == "standard") {

  z <- qnorm(p = 1 - alpha / 2)
  se <- apply(m_hat_star, 2, sd)
  lwr <- m_hat - z * se
  upr <- m_hat + z * se

} else if (type_CI == "quantiles") {

  q <- apply(m_hat_star, 2, quantile, probs = c(alpha / 2, 1 - alpha / 2))
  lwr <- q[1, ]
  upr <- q[2, ]

} else {
  stop("Incorrect type_CI")
}
# Return evaluation points, estimates, and confidence intervals
```

```
    return(data.frame("exdat" = exdat, "m_hat" = m_hat, "lwr" = lwr, "upr" = upr))
}
```

Finally, we compare and validate the correct behavior of the confidence intervals, for the two specifications of `type_boot` and the two types of perturbations. For this purpose, we simulate the following sample of size $n = 100$ from the regression model $Y = m(x) + \epsilon$ where $m(x) = 0.25x^2 - 0.75x + 3$, with $X \sim \mathcal{N}(0, 1.5^2)$ and $\epsilon \sim (0, 0.75^2)$. Figure 5 shows the simulated sample. In this case the simulated observations are concentrated in the left side of the plot.

Simulate observations m(x)
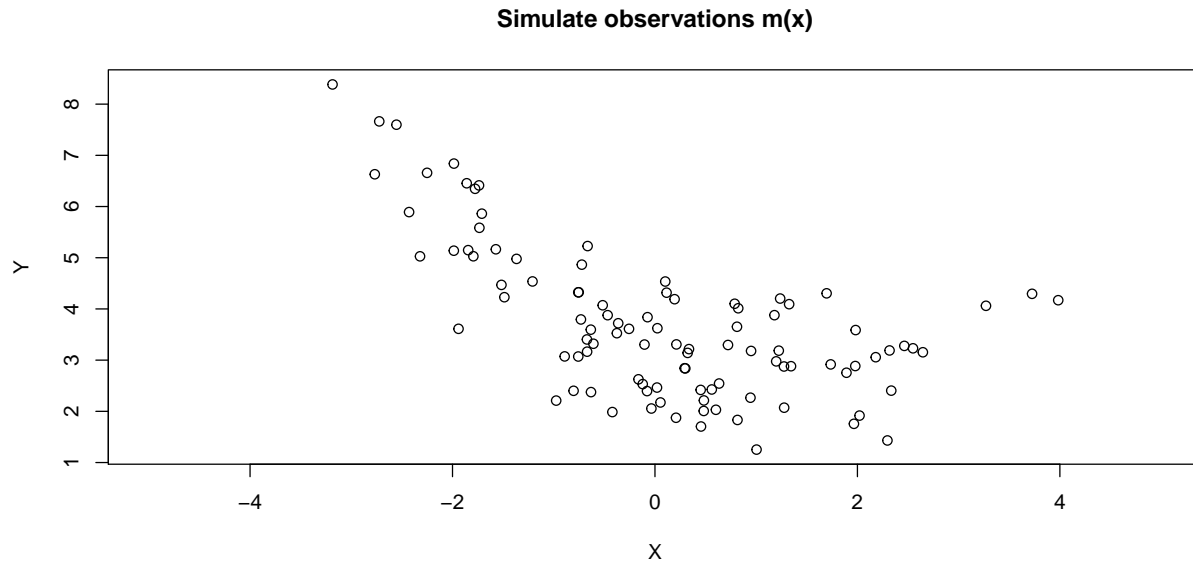
Figure 5: Simulation of 100 observations for m(X). Seed = 12345

Then, we fit a model using the function `npregbw()` with `regtype = "lc"` and the final model is created with the function `npreg()`. For reference, figures 6 and 7 shows the confidence interval under normal approximation and quantile compute by `np::npplot`. In particular, we focus on the right side of the confidence intervals. In the case of quantile confidence interval (figure 7) the upper confidence bound is relatively close to the fitted regression $\hat{m}(X_i; q, h)$
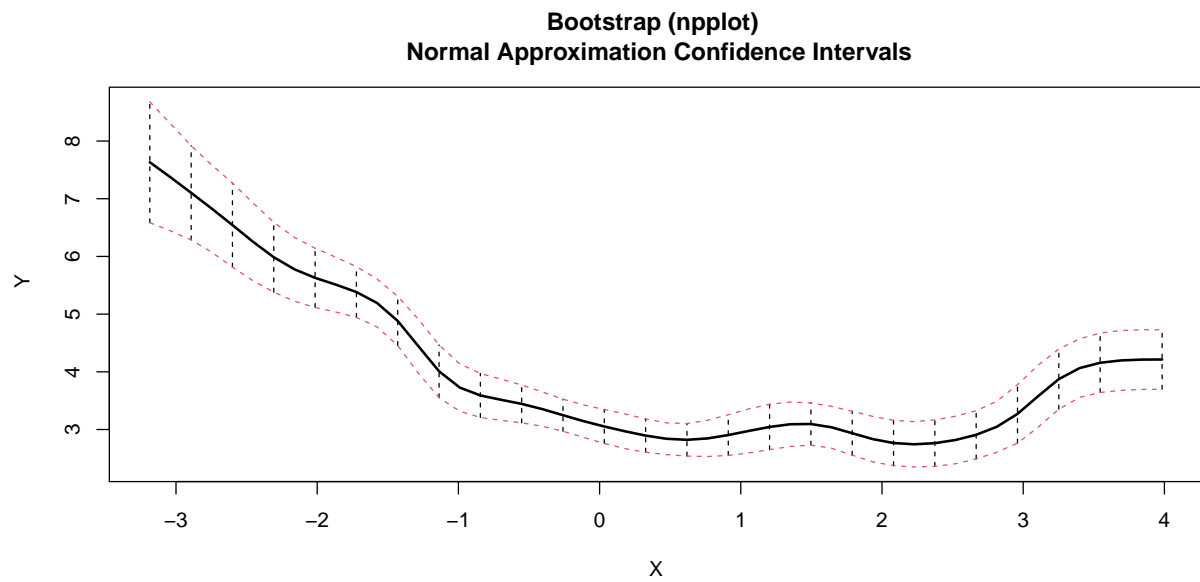
10

Bootstrap (npplot)
Normal Approximation Confidence Intervals

Figure 6: Bootstrap intervals: Default option npplot

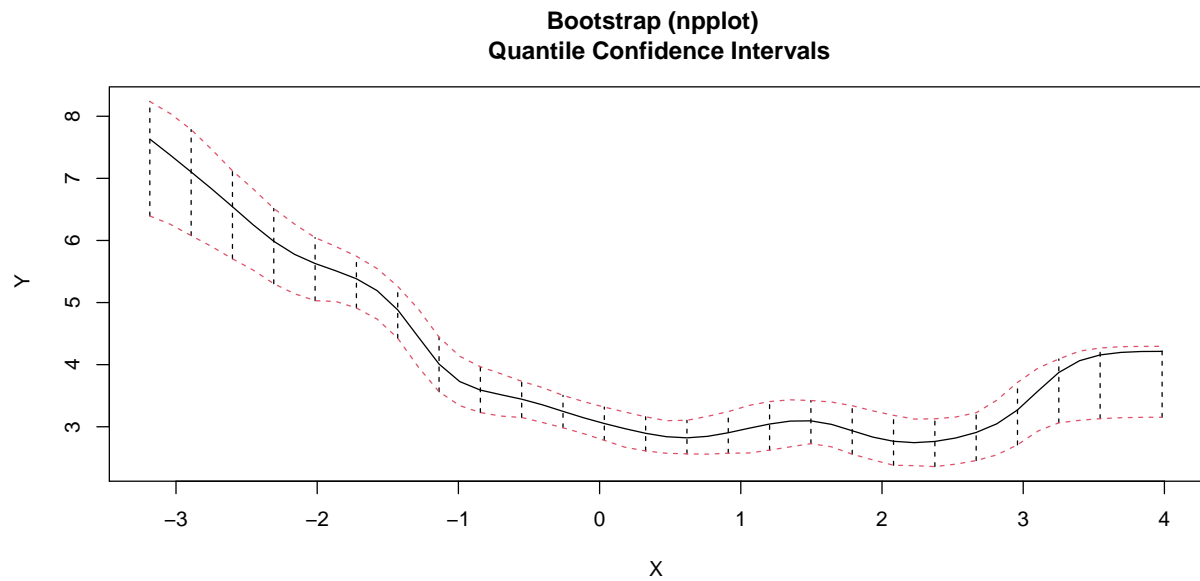**Bootstrap (npplot)**
Quantile Confidence Intervals

Figure 7: Bootstrap intervals: Default option npplot

Then, we compute 95% confidence intervals for $m(x)$ along x <- seq(-5, 5, by = 0.1). Figure 8 shows the case of the naive bootstrap. In this case, we see how the confidence intervals of the default quantile option of np::npplot coincide perfectly with the confidence interval given by np_pred_CI with the option type_bot = naive. Additionally, the light blue and green lines represents the extension of the predicted values for $m(x)$ in the grid between -5 and 5.

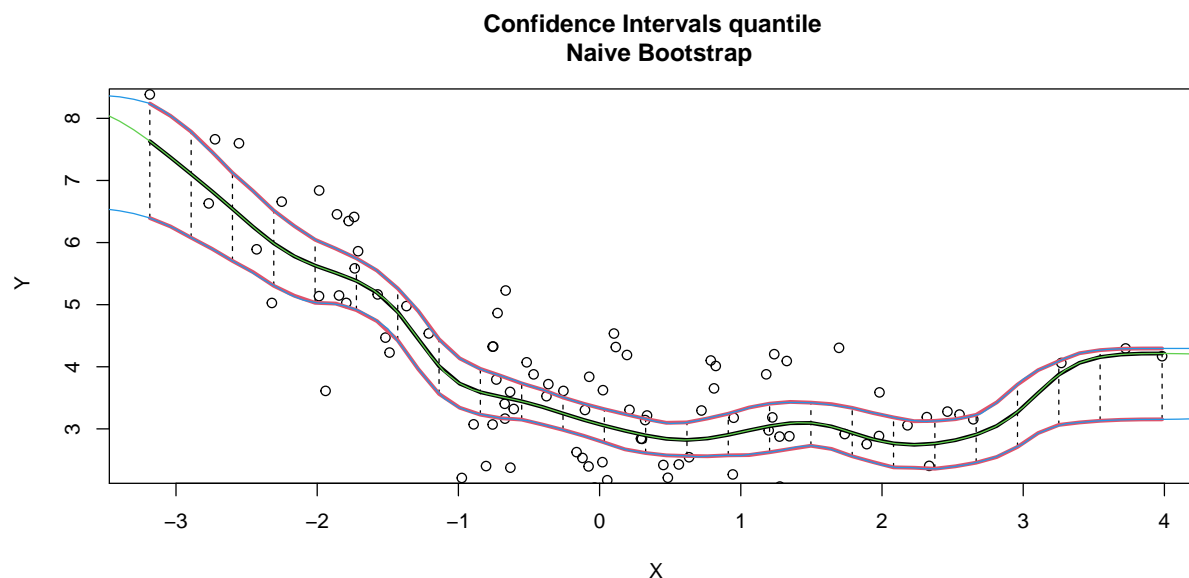

Figure 8: Function np_pred_CI: Naive Bootstrap

Finally, figures 9 and 10 shows the confidence intervals in case of wild bootstrap with normal and golden section perturbation. In this case, we notice how the confidence interval is in the function of the number of simulated observations. In the case of the tails, especially on the right side, the confidence intervals are wider in relation to the confidence intervals given by the naive bootstrap. On the other hand, in the sections of the graph where there is a concentration of simulated observations, the fitted lower and upper confidence intervals are similar for the two bootstrap methodologies.
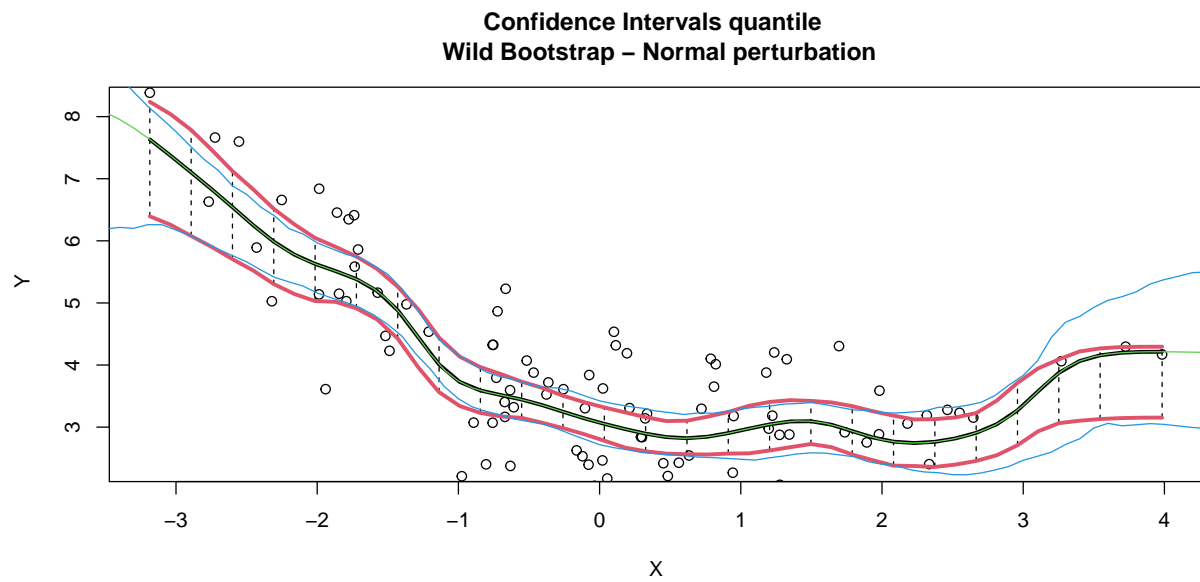
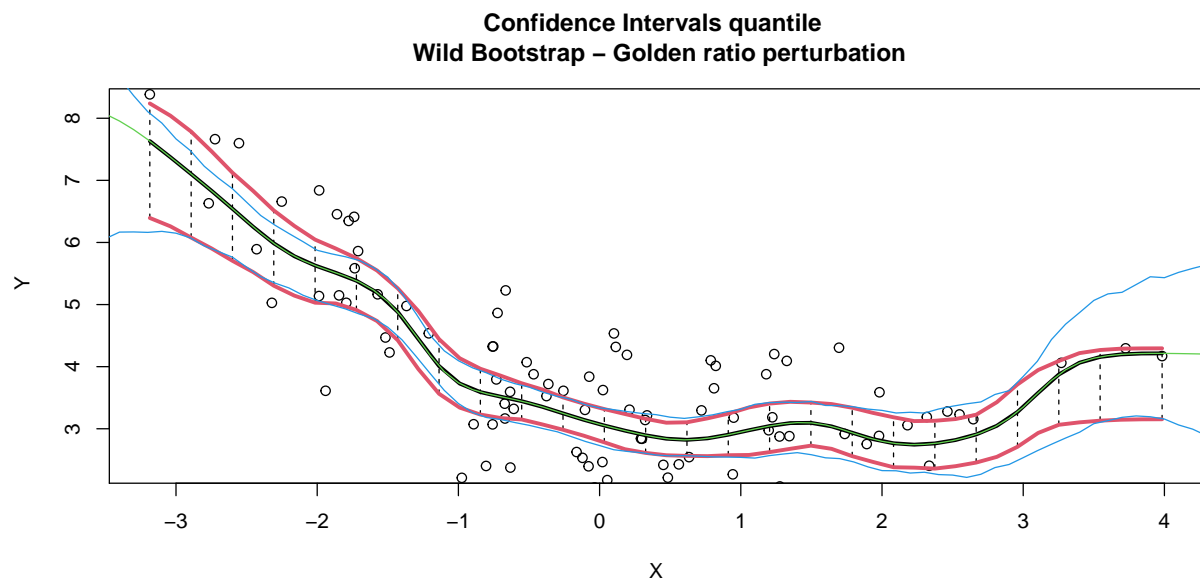Figure 9: Function np_pred_CI: Wild Bootstrap and normal perturbation

Figure 10: Function np_pred_CI: Wild Bootstrap and golden error perturbation