# KDE-debugging-demo

```r
# takes a n x 1 vector of data calculates silverman's rule of
# thumb for sigma
# https://github.com/statsmodels/statsmodels/blob/main/statsmodels/nonparametric/bandwidths.py
silverman <- function(x) {

    # apply and return silverman's rule of thumb
    sigma <- (0.9 * min(sd(x), IQR(x)/1.35) * length(x)^(-0.2))
    return(sigma)
}

# takes a point z, a vector of n means, and scalar sigma returns
# the density of z under the density of a mixture of Gaussians,
# each centered at mu with standard deviation sigma
phi0_k.z <- function(z, mu, sigma) {

    # calculate and return the density of z
    return((1/length(mu)) * sum(dnorm(z, mu, sigma)))
}

# vectorized version of phi0_k - takes a vector Z of length N,
# vector of n means and a scalar sigma Returns a vector of N
# densities
phi0_k <- function(Z, mu, sigma) {
    return(sapply(Z, phi0_k.z, mu = mu, sigma = sigma))
}

# function that takes a data matrix X and returns a bandwidth
# for each of the subjects (vector of length n)
get_bandwidths <- function(X) {

    # get dimensions of X
    n <- nrow(X)
    p <- ncol(X)

    # find the component-wise density for each of the
    # individuals also find silverman's rule of thumb for each
    # of the columns of X
    densities <- matrix(NA, n, p)
    sigma <- rep(NA, p)
    for (j in 1:p) {

        # find the value of sigma corresponding to the j-th
        # predictor
        sigma[j] <- silverman(X[, j])

        # calculate the resulting densities
```

```r
        densities[, j] <- phi0_k(X[, j], X[, j], sigma[j])
    }

    # calculate the harmonic mean for the sigma^2
    H <- 1/mean(1/sigma)

    # calculate the square root of the row-wise product of the
    # densities
    rowProds_sqrt <- rep(NA, n)
    for (l in 1:n) {
        rowProds_sqrt[l] <- prod(sqrt(densities[l, ]))
    }

    # return the final bandwidths
    return(H/rowProds_sqrt)
}
```

```r
# generate some data and try it out
set.seed(1)
n <- 2000
p <- 5
limits <- matrix(1:p, p, 2) * matrix(c(-1, 1), p, 2, T)
X <- matrix(runif(n * p, limits[, 1], limits[, 2]), n, p, T)
summary(X)
```

```
##        V1                  V2                 V3                   V4
##  Min.   :-0.997946   Min.   :-1.99727   Min.   :-2.999276   Min.   :-3.99840
##  1st Qu.:-0.486186   1st Qu.:-1.00985   1st Qu.:-1.488389   1st Qu.:-2.02589
##  Median :-0.016691   Median :-0.01738   Median :-0.006958   Median :-0.11502
##  Mean   :-0.005323   Mean   :-0.01006   Mean   : 0.001604   Mean   :-0.04986
##  3rd Qu.: 0.507396   3rd Qu.: 1.02458   3rd Qu.: 1.553405   3rd Qu.: 1.94929
##  Max.   : 0.997820   Max.   : 1.99305   Max.   : 2.995168   Max.   : 3.99945
##        V5
##  Min.   :-4.99894
##  1st Qu.:-2.37765
##  Median : 0.09663
##  Mean   : 0.11980
##  3rd Qu.: 2.70045
##  Max.   : 4.99455
```

```r
# demo the functions:
library(ggplot2)

# silverman rule of thumb for each column of X:
apply(X, 2, silverman)
```

```
## [1] 0.1138329 0.2312477 0.3442222 0.4513766 0.5785286
```

```r
# different n to try
(n_j <- seq(200, n, 200))
```

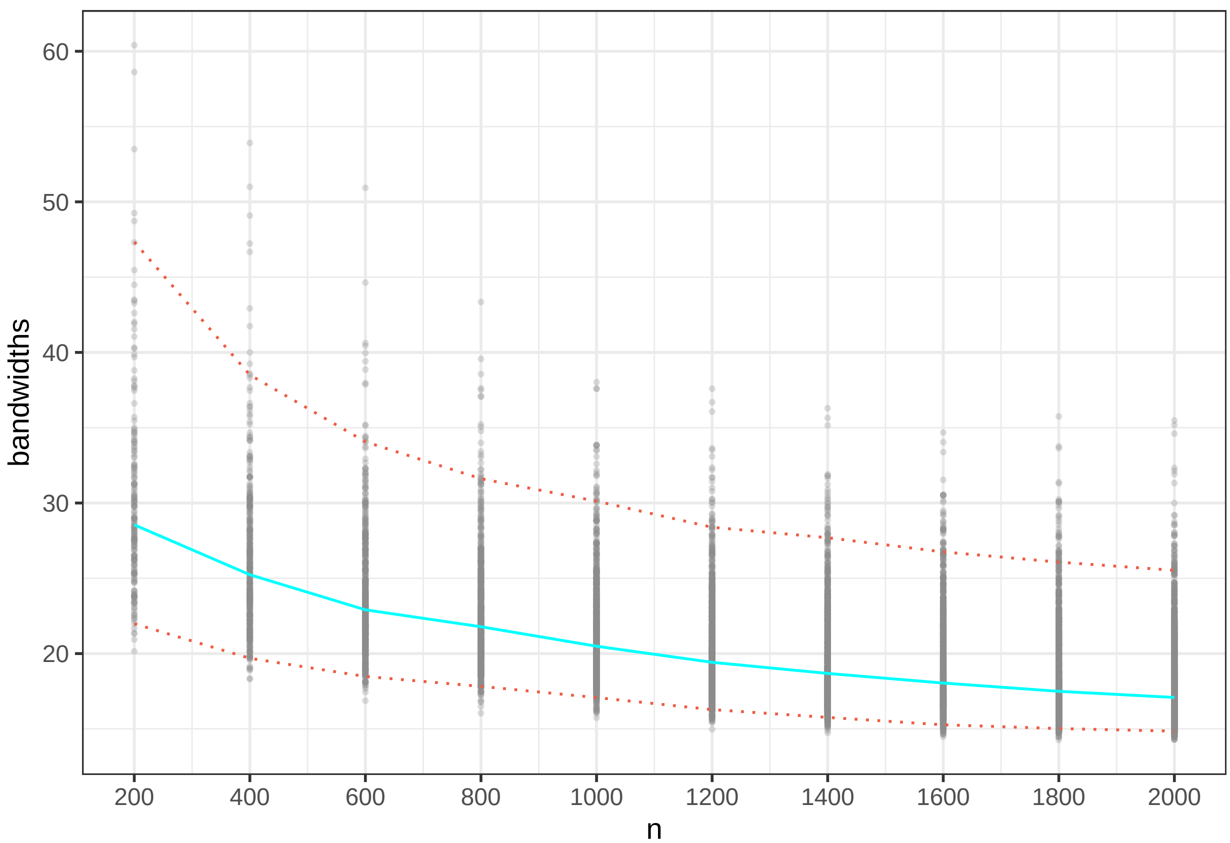
```
##  [1]  200  400  600  800 1000 1200 1400 1600 1800 2000
```

```r
# matrices for storing quantiles and bandwidths
bandwidth.dist <- matrix(NA, 3, length(n_j))
colnames(bandwidth.dist) <- paste("n =", n_j)
rownames(bandwidth.dist) <- c("0.025", "0.5", "0.975")
bandwidths <- vector("list", length(n_j))

# plot the distribution of the bandwidths for varying p
bandwidth_dist_graph <- ggplot()

for (j in 1:length(n_j)) {
    bandwidths[[j]] <- get_bandwidths(X[1:n_j[j], ])
    bandwidth.dist[, j] <- quantile(bandwidths[[j]], c(0.025, 0.5,
        0.975))
    bandwidth_dist_graph <- bandwidth_dist_graph + geom_point(data = cbind.data.frame(n = n_j[j],
        bandwidths = bandwidths[[j]]), aes(n, bandwidths), size = 0.5,
        color = "gray55", alpha = 0.3)
}

bandwidth_dist_graph + theme_bw() + geom_line(data = data.frame(n = n_j,
    bandwidths = bandwidth.dist[1, ]), aes(n, bandwidths), color = "tomato2",
    linetype = "dotted") + geom_line(data = data.frame(n = n_j, bandwidths = bandwidth.dist[3,
    ]), aes(n, bandwidths), color = "tomato2", linetype = "dotted") +
    geom_line(data = data.frame(n = n_j, bandwidths = bandwidth.dist[2,
        ]), aes(n, bandwidths), color = "cyan") + scale_x_continuous(labels = n_j,
    breaks = n_j)
```



```r
bandwidth.dist
```

```
##        n = 200  n = 400  n = 600  n = 800 n = 1000 n = 1200 n = 1400 n = 1600
## 0.025 21.98292 19.68691 18.48045 17.82194 17.07952 16.28048 15.76098 15.27032
## 0.5   28.55039 25.23807 22.90742 21.78216 20.48809 19.42012 18.68163 18.04493
## 0.975 47.33889 38.50804 34.06493 31.61209 30.12157 28.38993 27.69623 26.75168
##       n = 1800 n = 2000
## 0.025 15.02095 14.84863
## 0.5   17.49330 17.08638
## 0.975 26.07434 25.53007
```