Series 3

1. In this exercise we generate artificial data according to the model $Y_i = m(x_i) + \epsilon_i$, $i = 1, \ldots, 101$,

$$m(x) = x + 4\cos(7x)$$

$\epsilon_1, \ldots, \epsilon_{101}$ are i.i.d. $\mathcal{N}(0, 1)$. In a) and b) we consider the situation with equidistant x_i. In c) we are using non-equidistant x_i.

a) Carry out a simulation where you simulate data according to the model above 1000 times. Use 101 equidistant x_i between -1 and 1.

```
> x <- seq(-1, 1, length = 101)
```

For each dataset compute the Nadaraya-Watson, the Local Polynomial and the Smoothing Splines regression estimators at every $x_i, i = 1, \ldots, 101$. Save the results of each estimator in a matrix with rows being x-positions and columns simulation runs. For the Nadaraya-Watson estimator, use a bandwidth of 0.2. To get (approximately) the same degrees of freedom for the two other estimators, use `span = 0.2971339` for the Local Polynomial estimator (`loess`) and `spar = 0.623396` for the Smoothing Splines estimator (`smooth.spline`). **R-Hints:**

```
> set.seed(79)
>   ## nw = Nadaraya-Watson, lp = Local Polynomial, ss = Smoothing Splines
>   estnw <- estlp <- estss <- matrix(0, nrow = 101, ncol = nrep)
>   for(i in 1:nrep){
     ## Simulate y-values
     y <- m(x) + rnorm(length(x))
     ## Get estimates for the mean function m(x) in the points given by the vector x
     estnw[,i] <- ksmooth(x, y, kernel = "normal", bandwidth = 0.2, x.points = x)$y
     estlp[,i] <- predict(loess(....), newdata = x)
     estss[,i] <- predict(smooth.spline(....), x = x)$y
   }
```

At each position x_i compute the empirical bias of $\hat{m}(x_i)$ (mean over all simulations minus *true value*), the variance of $\hat{m}(x_i)$, and the mean square error (MSE) of $\hat{m}(x_i)$ for each of the estimators. Plot each of these three quantities against x_i for all three estimators (to get one graph for bias, one for variance, one for MSE). If you save each of these quantities (bias, variance, MSE) in a 101×3 matrix you can do the plots with `matplot`. Use `apply` to get the means and the variances.

What is the connection between the bias and the curvature $m''(x)$? How does the bias behave at the boundary?

b) Calculate the corresponding estimated standard error as in formula (3.7) on p. 29 in the script for each simulation run, x-value and estimator. To manually calculate the estimated standard errors we need the corresponding hat matrices (see p. 28 in the script). We can easily get them by using linear algebra. If S is the hat matrix, the j^{th} column is given by Se_j, where e_j is the j^{th} standard basis vector. The hat matrices only depend on the design points x_i and they do *not* have to be calculated for each simulation run. For the Nadaraya-Watson kernel estimator, for instance, you can calculate the hat matrix as follows.

```
> Snw <- matrix(0, nrow = 101, ncol = 101)
> In <- diag(101) ## identity matrix
> for(j in 1:101){
   y <- In[,j]
   Snw[,j] <- ksmooth(x, y, kernel = "normal", bandwidth = 0.2, x.points = x)$y
 }
```

To calculate estimated standard errors, you can then use your script file from a) adding the following commands to the for-loop (see also p. 29 in the script for the formula for the estimator of the error variance σ_ε^2):

```
> sigma2nw <- sum((....)^2) / (length(y) - sum(diag(Snw)))
> senw[,i] <- sqrt(sigma2nw * diag(....))
```

Note that sum(diag(Mat)) calculates the trace of a matrix Mat. Matrix multiplication is done using $\%*\%$ in R. You may also want to consider crossprod() or tcrossprod().

How many times (out of 1000 simulations) does the pointwise confidence interval at $x = 0.5$ contain the true value $m(0.5)$, i.e., what is the so-called "coverage rate"? For the construction of the pointwise confidence interval see p. 29 in the script. How often does the confidence band for all points x_i *simultaneously* contain *all* true values?

c) Repeat a) and b) but with non-equidistant x-points. Use the R-commands

```
> set.seed(79)
> x <- sort(c(0.5, -1 + rbeta(50, 2, 2), rbeta(50, 2, 2)))
```

to generate the points. You can use `rug(x)` to visualize the distribution in the plots in a) and b). Again, for the Nadaraya-Watson estimator, use a bandwidth of 0.2. To get the same degrees of freedom you should now use `span = 0.37614` in `loess` and `spar = 0.79424` in `smooth.spline`.

Preliminary discussion: Friday, March 13.

Deadline: Friday, March 20.