Series 3

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

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

$\epsilon_1, \dots, \epsilon_{101}$ 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 a 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 $x_i, i = 1, \dots, 101$. Save the results of each estimator in a matrix with rows being x-positions and columns simulation runs. To get (approximately) the same degrees of freedom use `span = 0.2971339` for `loess` and `spar = 0.623396` for `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
    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 (mean over all simulations minus *true value*) and variance. Plot these quantities against x_i for each estimator. If you save each of these quantities in a 101×3 matrix you can do the plots with `matplot`. Use `apply` to get the means and the variances.

b) Calculate the corresponding estimated standard error for each simulation run, x-value and estimator. To manually calculate the estimated standard errors we need the corresponding hat matrices (see lecture notes). 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:

```
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 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"? How often does the confidence band for all points *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). To use 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 18.

Deadline: Friday, March 25.