Series 4

1. In this exercise, we model the log-returns of the BMW stock (business-daily, between June 1986 and March 1990). The log-returns are defined as follows:

$$X_t = \log\left(\frac{P_t}{P_{t-1}}\right),$$

where P_t is the stock price at time t. Log-returns can be modelled by

$$X_t = \sigma_t \epsilon_t, \text{ where } \mathbf{E}[\epsilon_t] = 0, \mathrm{Var}(\epsilon_t) = 1, \tag{1}$$

ϵ_t independent of $\{X_s;\ s < t\}$, $\sigma_t^2 = v(X_{t-1})$, where $v:\ \mathbb{R} \mapsto \mathbb{R}^+$ is the so-called "volatility function". Thus, X_t depends on $\{X_s;\ s < t\}$ only through X_{t-1} (Markov-property).

The model can be fitted by nonparametric regression of the function v in

$$Y_t = X_t^2 = v(X_{t-1}) + \eta_t, \text{ where } \eta_t = \sigma_t^2(\epsilon_t^2 - 1)$$

is treated as error term.
Note: Other usual model assumptions on errors, such as independence, are not fulfilled by η_t, but with some effort (don't try!), it can be shown that v can be optimally estimated by the same estimation methods as if the η_t were independent errors.

a) The `bmw` data set contains the *returns* and not the *log-returns* of the BMW stock. It is difficult to find the original price, but for the sake of the exercise, let us assume that $P_0 = 40$, which is a reasonable value and should not have too much influence on the results. Given this information and the time series of returns, construct the time series of log-returns.

```
> bmwr <- scan("http://stat.ethz.ch/Teaching/Datasets/bmw.dat")
```

b) Fit the data using the nonparametric regression methods Nadaraya-Watson, Local Polynomial and Smoothing Splines for the regression function v.
Comment on the results, and compare the fits obtained using the mentioned nonparametric estimators. Look at the estimated volatility function as a function of X_t and at the estimated implied volatility as a function of time.
R-hint: Use `loess` for Local Polynomial, `smooth.spline` for Smoothing Splines and `ksmooth` for Nadaraya-Watson kernel regression.
The methods have no consistent way to choose the *degree of smoothness*. One can do that via cross-validation (see next problem sheet). For the present series, however, we give you the parameters to use. For `loess`, the smoothness is defined by the parameter `span`, which indicates the fraction of data to include in the support of the kernel (expressed as a number between 0 and 1). For the exercise, you can use the default value of 0.75, but try to play with it and see how it influences the results.
For `smooth.spline`, you can define the smoothness in terms of equivalent degrees of freedom (edf), which can be computed as the trace of the hat matrix (see lecture notes for more details). Again you can play with different values, but for the sake of comparison, you can use the same edf as in `loess`, which can be recovered from the output by `fit$trace.hat`.
For `ksmooth`, the smoothness is defined in terms of the bandwidth. Try to play with different values, and see how it influences the result. To ensure that you have the same smoothness as for the other methods, you would need to numerically search for the bandwidth value that results in the same edf. To save you this trouble, here is the answer: use `h=0.16`.
Remark: `ksmooth` internally reorders its x input in increasing order, so you will lose the time ordering. To recover it, you have to do the following:

```
ox <- order(x)
fit <- ksmooth(x,y,...)
fit$x <- fit$x[order(ox)]
fit$y <- fit$y[order(ox)]
```

Check the model assumptions, but do not spend too much time on this since the structure of the data is pretty unclear. Note that for computing residuals, it is necessary to know the fitted values at the data points. For `ksmooth`, they are provided via argument `x.points`, and for `loess` and `smooth.spline`, they are provided via `fitted()`.

c) Fit the data using the functions `glkerns` (kernel regression with global optimal bandwidth) and `lokerns` (kernel regression with local optimal bandwidth) of the R package `lokern`. Compare the fits. Plot the local bandwidths from `lokerns`, and compare them to the global bandwidth of the function `glkerns`. How does the local bandwidth relate to the density of the data?
Remark: It is not so easy to control how the function optimizes the bandwidth internally, and so this can easily lead to misleading results. In the present case, be careful to pass the argument `is.rand=TRUE` to specify that the design is not fixed and `hetero=TRUE` for heteroscedastic errors.

2. In this task, our goal is to understand the difference between *interaction* and *correlation*.
Note that any two correlated variables are in particular not independent.

a) Define what it means for two (random) variables X_1 and X_2 to be *correlated*. Also explain what *interaction* between X_1 and X_2 means in relation to the outcome Y.

b) X_1 is now a continuous variable, and X_2 is a variable taking only two values (that is, a factor variable with two levels). The dependent variable is Y. Draw by hand a sketch for the following situations:

1. X_1 and X_2 are uncorrelated, and there is no interaction between X_1 and X_2.
2. X_1 and X_2 are correlated, and there is no interaction between X_1 and X_2.
3. X_1 and X_2 are uncorrelated, and there is interaction between X_1 and X_2.
4. X_1 and X_2 are correlated, and there is interaction between X_1 and X_2.

Note that X_2 needs to take real values as well in order to be able to calculate its expectation and its correlation with the other variable. Hence, we implicitly assume here that the two levels of X_2 are real numbers.

Preliminary discussion: Friday, March 25.

Deadline: Friday, April 01.