5. Adding Noise

- Adding noise is a perturbative protection method for microdata, which is typically applied to continuous variables.
- **Motivation:** This approach protects data against exact matching with external files if, for example, information on specific variables is available from registers.
- While this approach sounds simple in principle, many different algorithms can be used to overlay data with stochastic noise.
- It is possible to add uncorrelated random noise. In this case, the noise is usually distributed and the variance of the noise term is proportional to the variance of the original data vector.

5. Adding Noise

- **Important** - Adding uncorrelated noise preserves means, but variances and correlation coefficients between variables are not preserved.
- This statistical property is respected, however, if correlated noise method(s) are applied.
- For the correlated noise method , the noise term is derived from a distribution having a covariance matrix that is proportional to the co-variance matrix of the original microdata.
- In the case of correlated noise addition, correlation coefcients are preserved and at least the co-variance matrix can be consistently estimated from the perturbed data.

5. Adding Noise

Implementation

- The data structure may differ a great deal, however, if the assumption of normality is violated.
- Since this is virtually always the case when working with real-world datasets, a robust version of the correlated noise method is included in **sdcMicro**.
- Noise can be added to variables in **sdcMicro** using function `addNoise()` or by using sdcMicroGUI.
- More information can be found in the help le by calling `?addNoise` or using the graphical user interface help menu.

5. Adding Noise

Outliers / Implementation

- In **sdcMicro**, several other algorithms are implemented that can be used to add noise to continuous variables. For example, it is possible to add noise only to outlying observations.
- In this case, it is assumed that such observations possess higher risks than non-outlying observations.
- Other methods ensure that the amount of noise added takes into account the underlying sample size and sampling weights.

6. Shuffling

- **Mathematically Complex** Various masking techniques based on linear models have been developed in literature, such as multiple imputation, general additive data perturbation and the information preserving statistical obfuscation syrithetic data generators.
- These methods are capable of maintaining linear relationships between variables but fail to maintain marginal distributions or non-linear relationships between variables.
- Several methods are available for shuffling in **sdcMicro** and **sdcMicroGUI**, whereas the first (default) one (ds) is recommended to use.

6. Shuffling

-
- Shuffling simulates a synthetic value of the continuous key variables conditioned on independent, non-condential variables.
- After the simulation of the new values for the continuous key variables, reverse mapping (shuffling) is applied.
- This means that ranked values of the simulated values are replaced by the ranked values of the original data (columnwise).

6. Shuffling

- **Example** To explain this theoretical concept more practically we can assume that we have two continuous variables containing sensitive information on income and savings.
- These variables are used as regressors in a regression model where suitable variables are taken as predictors, like age, occupation, race, education.
- Of course it is of crucial to find a good model having good predictive power.

6. Shuffling

- New values for the continuous key variables, income and savings, are simulated based on this model.
- However, these expected values are not used to replace the original values.
- Rather a value re-assignment system (shuffling) of the original values using the generated values is carried out.

6. Shuffling

Value Assignment System

- This approach (reverse mapping) is applied to each sensitive variable can be summarized in the following steps:
    1. rank original variable
    2. rank generated variable
    3. for all observations, replace the fitted value of the variable with rank i with the value of the original sensitive variable with rank *i*.
    4. once finished, the modied variable contains only original values and is finally used to replace the original sensitive variable.

6. Shuffling

- It can be shown that the structure of the data is preserved when the model fir is of good quality.
- In the implementation of sdcMicro, a model of almost any form and complexity can be specified (see `?shuffling` for details).