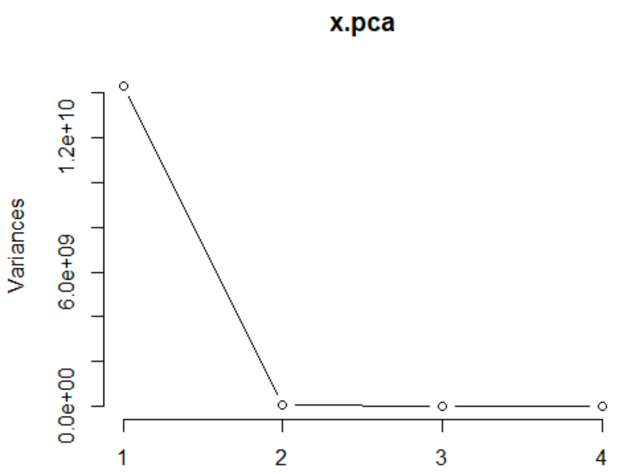
Explanations for Exercise 7

In task 1, the idea of principal components is that instead of original variables Var1,…,Var4 you'll get 4 principal components, which each contain a linear combination of all 4 original variables and that linear combination captures the variance in the Variable space into orthogonal directions. First principal component (PC1) captures the highest variance and it gets smaller for each following component. So, when you run 'script_PCA_E7.R', you'll get the output:

```
Rotation (n x k) = (4 x 4):
               PC1           PC2           PC3           PC4
Var1 -2.306577e-05  1.185786e-03 -9.999895e-01 -4.427279e-03
Var2  4.770627e-02  9.988607e-01  1.183337e-03  2.167368e-06
Var3 -3.746301e-07 -3.057227e-06  4.427278e-03 -9.999902e-01
Var4 -9.988614e-01  4.770621e-02  7.960712e-05  5.808034e-07
```

where you'll see the linear combinations for each variable. The numbers inside the PC tell which variables are the dominant ones in this PC. You cannot compare the values between PC:s, and you do not use this information for selecting, how many components you may need. Absolute value of the number is considered here, when interpreting, which variables dominate, as the association can be positive or negative.

Then you get the plot



where you have the variance of each PC presented visually.

Then you get the importance analysis for each component. As you remember from the class, the PC:s are ordered based on their variance value. Here in the first row, you have standard deviation, which is sqrt(var). So PC1:s std dev is squared and you get the value in the upper picture (y-axis). Here it is difficult to see as the deviations for PC2-4 are so small, but you can verify this in the next step, where you scale the variables. Proportion of the variables tell, how much compared to the total variance, you get with each PC. And the cumulative variance is cumulative sum of row 2.

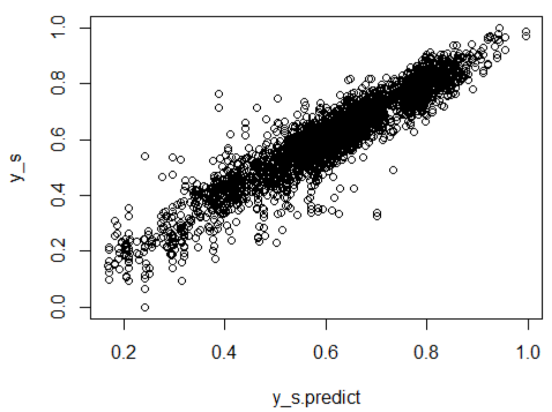
```
Importance of components:
                             PC1       PC2   PC3   PC4
Standard deviation     1.197e+05 6.207e+03 45.49 0.467
Proportion of Variance 9.973e-01 2.680e-03  0.00 0.000
Cumulative Proportion  9.973e-01 1.000e+00  1.00 1.000
```

In task 2 your goal is to make a model that can predict the values of y with the input variables in X. You can compare the situation to linear regression, but here the neural network is capable for finding a highly nonlinear functional presentation of variables in X. In real life, the systems can be very complex and you need to use automatic methods for finding the function. Neural networks are called “the black box method”, because you cannot see, how the relationship between X and y is built.

You will learn more about these kinds of methods in Introduction to artificial intelligence and Machine learning –courses. In this course, the idea is not to teach you prediction methods, but it is difficult to demonstrate pre-processing without using it somewhere, and that’s why this model was included to this exercise.

So you built a model to predict the values of y. You could use this prediction model later to predict y, if you did not know it, but only the variables X. I have built models that predict the steel quality (impact toughness for example) based on the process parameters. So for new products, the user does not have to measure the quality, but instead, it can be predicted with this mode. And the user can test different process variable values and to check, how the model quality changes. In a way, this model is a digital twin of the product.

So, when you now run the script and it prints the correlation between original y and the predicted y, you’ll get the idea of how well did your model predict. Here the value 0.938 means that the correlation is quite high and the model is quite good. Note that the value can be a bit different every time, as the method is stochastic and your result will always change a bit.



Here are the scaled y values and scaled predicted values in one plot. The perfect match would produce a diagonal line from 0 to 1, but here the values are a bit scattered around that (invisible) line. The better the model, the more closely to that line the observations would be.