Roadmap of the Tutorial

- **Practical part (14.00-17.00):**
  - learning by doing

- go into the directory `/usr/local/matlab/bin`,
- start `matlab`
Generation and manipulation of vectors

- generate a row and column vector using
  - direction definition  `x=[1 3 8];`
  - using `ones, zeros, rand, randn` or the colon operator:
    in case of error or unknown commands use `>> help function`
- check the workspace content with `>> whos`
- check the size with `>> size(your_array)`
- transform your row vector into a column vector and then concatenate both.
• generate a square matrix of the same size as your column vector
  – use either `rand`, `randn`
• check the workspace content with `>> whos`
• check the size with `>> size(your_array)`
• multiply the column vector with the matrix and assign the result to a new variable
• take the inner product of the result and your original column vector (there are at least three ways to do that - how many do you find?)
Indexing matrices

- generate a random $3 \times 4$-matrix $A$
- access the elements $A(3, 1), A(1, 4)$ and assign zero to them
- what are the corresponding linear indices? (try it!)
- display the third column and the second row
- display all elements which are larger than 0.5
- generate a zero array of size $2 \times 3 \times 4$
  assign the matrix $A$ such that $B_{2kl} = A_{kl}$ with $1 \leq k \leq 3$ and $1 \leq l \leq 4$.
- display row and column indices of $A$ which are between 0.2 and 0.5.
Linear Equations and first functions

- use >> x=rand(30,2); y=0.5*x(:,1)+0.2*x(:,2);
- define >> H=x’*x and >> b=x’*y
- solve the linear system $Hw = b$,
- what are the coefficients $w$ ?
- (add a little bit of Gaussian noise to $y$ and repeat )
- write a function `LeastSquares` with input $x, y$ which has as output $w$.
- what will happen if $y$ has more than one column ?
- add a comment to the function - and check what is shown when you use `help LeastSquares`

For the fast ones:

- write a function `MatrixPower` with a symmetric matrix $A$ and a real number $c$ as input - compute $A^c$ using the eigenvalues and eigenvectors of $A$. 
JIT compilation versus vectorization

- implement functions $\text{PNorm1}$ and $\text{PNorm2}$ and repeat the experiment of the lecture
- check runtimes for the growing arrays once when you preallocate memory and once without
- use `clear` to free memory of variables
Debugging

- Download the file `dist_euclideanBUG.m` from the Matlab-Tutorial homepage.
- It is supposed to define the distances between two sets of points $X$ and $Y$; e.g., with $X=\text{rand}(30,6)$; $Y=\text{rand}(20,6)$; $D=\text{dist}_\text{euclideanBUG}(X,Y)$ computes the $30 \times 20$ distances between the 6-dimensional vectors.
- Use the debugger to find the bug in `dist_euclideanBUG.m`. 
Plotting

- reproduce the plots from the lecture
- play around with the parameters - different colors, plotting styles etc.
- generate other functions and plot them
Profiler

- generate data with
  \[
  \text{>> } x = \text{randn}(500, 2); \ x = [x \ ; \ \text{randn}(500, 2)+3]
  \]
- run
  \[
  \text{Ind} = \text{SpectralClustering}(x)
  \]
  in the profiler