math 1MP assignment 3

Ben Bolker

20:19 09 March 2017

Due Friday 17 March at midnight (the end of the day), in the Dropbox on Avenue to Learn. As usual, the answers *must* be submitted as a module (text file) called `yourmacid_hw3.py`, e.g. mine would be `bolker_hw3.py`. If your module contains extra information (e.g. web resources you used or other students you worked with), please put them as comments (i.e. preceded by `#`).

- All of your functions *must* have docstrings.
- For this assignment, *please try to avoid using `for` loops as much as possible*; the more you avoid `for` loops, the better you will learn `numpy`, the better you'll do on the `numpy` questions on the exam ... (It is assumed that you will be using `numpy` for all answers.)

1. Write a function `calc_frac(a,axis=0)` that takes a 2-dimensional numpy array containing only zeros and ones and returns an array giving the fraction of ones in each column (if `axis=0`) or in each row (if `axis=1`).
2. Write a function `calc_rel_frac(a,axis=0)` that takes a 2-dimensional numpy array and returns the proportion of ones in each column or row *divided by the column or row that has the smallest proportion of ones*. (As in the previous question, the `axis` argument controls whether the function should be evaluating columns [`axis=0`] or rows [`axis=1`].) If there are any columns/rows that are made up entirely of zeros, the program should raise a `ValueError`.
3. Write a function, `check_symmetric(a,tol=1e-8)` that takes a 2-dimensional square array (i.e., a matrix) and returns a boolean value that reflects whether `b` is symmetric, i.e. that `a.transpose()` is equal, within tolerance `tol`, to `a`. Make sure to:

- raise a `ValueError` if the array is not square (number of rows != number of colums)
- use an appropriate test for floating-point near-equality; that is, return `True` if, for every i and j, $\text{abs}(A_{ij} - B_{ij}) < \text{tol}$, and `False` otherwise.

4. The *coefficient of variance* of a vector `x` is the ratio of its standard deviation to its mean. Define a function `arg_cvmax(a,axis=0)` that computes the coefficient of variation of each column or row of a 2-dimensional array and returns the *index* of the column or row with the maximum coefficient of variation (hint: the `.argmax` array method will probably be useful ...)