Fortran and Python HW5

Be sure to read the language-specific notes before you start. When you have completed the homework, upload all source files, paste any standard output into the text box, and upload images of your plots.

Create two modules. The first module should be called units and should contain the following functions:

F_to_C

C_to_F

C_to_K

K_to_C

F_to_K

K_to_F

inch_to_cm

cm_to_inch

cm_to_m

m_to_cm

ft_to_inch

inch_to_ft

lb_to_kg

kg_to_lb

The first 6 convert among Fahrenheit, Celsius, and Kelvin temperature scales. The purpose of the other functions should be obvious. They will be very short (in Python two-three lines each—Fortran programmers will have more typing to do). Use conversion factors accurate at least to two decimal places (you can get the correct factors from Google).

Create another module called stats which will contain a few functions (the number will depend on your language). The module will contain a global variable called MISSING which will be given the value of

-99.

The first procedure in stats should be called reject_outliers(A). It will use the Chauvenet criterion to reject outliers in the array of data A, which is rather crude but will work for our purposes. Here is the algorithm:

1. Get and store the mean and standard deviation of A as well as its length.
2. criterion=1/(2.*length(A))
3. Compute the array of absolute values of deviations devs=abs(A-mean(A))/std_dev(A)
4. Divide the devs array by sqrt(2.)
5. Compute the probability array probs=erf(devs)
   Where erf() is a special function called the error function
6. Compute an array valid whose elements are True/.true. if probs(i)>=criterion or otherwise is False/.false.
7. Return valid
8. The array valid can be used to mask the array of data, or any corresponding arrays in the data (i.e. extract outliers).

The second procedure should look through an array A for missing data. If it finds an element that has the MISSING value, then if it is the first element of A substitute the next value in the array, otherwise substitute the previous value. The array A should be changed in place (i.e. in the same variable). Return the number of missing values found. This is a particularly stupid way to fix data but that is not of any concern to us right now.

Now write two programs (don't worry, they will be short). You will use your two modules for both programs.

Program 1. Download VARICHMO.txt from Resources/Data/FP_HW. The columns are month, day, year, and average temperature for the 24 hours. You don't need the date columns, just use any method you wish to acquire the temperature data. Plot the temperatures before and after "cleaning" the data. (First run it through fix_missing, then run it through reject_outliers). Plot the temperature data before cleaning it. After cleaning it, convert the temperatures from Fahrenheit to Kelvins and plot it. Print the number of missing values. Print the number of missing values and the number of values rejected as outliers (either might be zero).

Program 2. Download bodyfact.csv from the same place. This dataset has a header; be sure to read it and note the units specified. For this one you do need more than one column. Store each column into an array with an appropriate name. Use the data to compute an array of BMIs. Make a scatter plot of the BMI versus percent body fat. Clean the BMI data as for the temperature data in Part 1. Print the number of missing values and the number of rejected values. Make a scatter plot of the cleaned data (BMI versus percent body fat again).

Language-Specific Notes:

Python: Use NumPy arrays. Do your computations in reject_outliers as array operations. You may use built-in numpy functions such as mean() and std(). You will need to use the numpy version of abs() in order to use it with an array operation. You will need to import the erf from a module called scipy.special and you should import only erf and erfc from it (erfc being another special function).

You can use the figure() method of pylab to make multiple figures and if you invoke show() only after you have prepared each figure, you can have both up at the same time. You can make PNG files upload from the image with the “Save As” menu item.

Fortran: as for Python, do computations in reject_outliers as array operations. All your built-in functions are already overloaded for arrays. The erf() is a new intrinsic in Fortran 2008 but it is already present even in the aged gfortran (4.4) on the Fir frontend, so you can use it. There are no mean or standard deviation intrinsics so you will need a few auxiliary functions in stats, but you may use the sum intrinsic. You will need to print out the data in an appropriate way to make plots, and you can use Excel or any other software that can plot it. In cases where you are doing a lot of before and after plotting, you may find it more convenient to write out a single CSV file (add a header to the file to help you remember what’s what), read that into Excel or whatever, and select columns to plot.   If you are using makemake you can have all the files in the same directory, but you’ll need to separate the two different executables by some means. If you want to have two Makefiles you can call one make.bmi and the other make.temps (or whatever you wish) and then you can build them separately with

```
make  -f  make.bmi

make  -f  make.temps
```