**Michelle Maclennan**

**21 September 2020**

**ATOC7500 – Application Lab #2**

**Regression, Autocorrelation, Red Noise Timeseries**

**in class Monday/Wednesday September 21/23, 2020**

**Notebook #1 – Autocorrelation and Effective Sample Size using Fort Collins, Colorado weather observations**

**ATOC7500\_applicationlab2\_AR1\_Nstar.ipynb**

**LEARNING GOALS:**

1) Calculate the autocorrelation at a range of lags using two methods available in python (np.correlate, dot products)

2) Estimate the effective sample size (N\*) using the lag-1 autocorrelation

3) Evaluate the influence of changing the sampling frequency and the specified weather variable on the memory/redness of the data as quantified by the autocorrelation and N\*.

**DATA and UNDERLYING SCIENCE:**

In this notebook, you will analyze the memory (red noise) in weather observations from Fort Colins, Colorado at Christman Field. The observations are from one year, but are sampled hourly. The default settings for the notebook analyze the air temperature in degrees F sampled once daily (every midnight). But other standard weather variables and sampling frequencies can also be easily analyzed. The file containing the data is called christman\_2016.csv and it is a comma-delimited text file.

**Non-exhaustive Questions to guide your analysis of Notebook #1:**

1) Start with the default settings in the code. In other words – Read in the data and find the air temperature every 24 hours (every midnight) over the entire year. Calculate the lag-1 autocorrelation using np.correlate and the direct method using dot products. Compare the python syntax for calculating the autocorrelation with the formulas in Barnes. Equation numbers are provided to refer you back to the Barnes Notes. What is the lag-1 autocorrelation?

The lag-1 autocorrelation using np.correlate is 0.846, which is the same as the lag-1 correlation calculated using the direct calculation. The direct calculation using python syntax requires you to have the mean, standard deviation, and known length of the time series. Then, it generates two timeseries from the data at t=t1 and t=t2, which are offset by the selected lag time. The mean of the original time series is subtracted from t1 and t2. Then, to calculate the correlation the dot product of t1 and t2 is divided by the length of the time series (n) minus the lag (which generates the mean of the temperature anomaly), and then divided again by the variance. This is really similar to the equation in Ch.2 of the Barnes notes, in that Equation 68 is divided by the variance in the python code. For a lag of zero, the autocovariance in Equation 68 is equal to the variance of the time series (or the mean of the product of departures from the long-term mean). To calculate the autocorrelation, Equation 68 is normalized by the variance, which in the case of the python syntax refers to dividing by the variance.

2) Calculate the autocorrelation at a range of lags using np.correlate and the direct method using dot products. Compare the python syntax for calculating the autocorrelation with the formulas in Barnes. Equation numbers are provided to refer you back to the Barnes Notes. How does the autocorrelation change as you vary the lag from -40 days to +40 days?

Please see answer to question (1) for more details about the comparison between the python syntax and the autocorrelation formula np.correlate. When you take any lag from -40 days to +40 days, you are changing the denominator of the equation (n – lag). For a lag of 1, the autocorrelation yields r = 0.846, and the direct calculation yields r = 0.846. For a lag of 10, the autocorrelation yields r = 0.725, and the direct calculation yields r = 0.725. For a lag of 20, the autocorrelation yields r = 0.616, and the direct calculation yields r = 0.616. For a lag of 40, the autocorrelation yields r = 0.403, and the direct calculation yields r = 0.403. The correlation coefficients are symmetric about lag = 0, so the same correlations occur for the negative lag times. Essentially, the higher the lag the lower the correlation. However, unlike a dataset with purely white noise, in which the correlation would drop to zero beyond a lag time of 0, in this case the correlation stays moderately strong for a long time even though it decreases. This helps to indicate that there is red noise, or memory associated with the time series. Because the distribution is symmetric about zero, we can conclude that the memory also extends both back and forward in time.

3) Calculate the effective sample size (N\*) and compare it to your original sample size (N). Equation numbers are provided to refer you back to the Barnes Notes. How much memory is there in temperature sampled every midnight?

The original sample size has 366 values, but the effective sample size (or the effective number of independent samples) is only 31, meaning our time series has a lot of memory or red noise (enough to reduce our number of effective samples to less than one tenth of the original sample size). From Leith 1973, we can divide the time step (delta t = 1) by the negative of the logarithm of alpha (the autocorrelation) to determine the e-folding time of the time series (Te). For this we get Te = 5.97.

4) Now you are ready to tinker … i.e., make minor adjustments to the code with the parameters set in the code to see how your results change. *Suggestion: Make a copy of the notebook for your tinkering so that you can refer back to your original answers and the unmodified original code.* For example: Repeat steps 1-3) above with a different variable (e.g., relative humidity (RH), wind speed (wind\_mph)). Repeat steps 1-3) above with a different temporal sampling frequency (e.g., every 12 hours, every 6 hours, every 4 days). How do you answers change?

I decided to repeat the steps above using wind speed at a 6-hourly temporal resolution. The lag-1 autocorrelation yields r = 0.365, and the direct calculation similarly yields r = 0.365. Unlike the temperature time series, there is far less memory or red noise for wind speed. The correlation coefficient drops off sharply for any lag not equal to zero, but does exhibit small oscillations between lags of +- 10 to 40 time steps (the plot label says Days but here we are using 6-hourly time steps). The effective sample size is only 681 compared with the original sample size of 1464, but this is a greater proportion than we saw with temperature.

**A screenshot of a cell phone

Description automatically generated**

**Notebook #2 – Red noise time series generation, Regression, and Statistical Significance Testing While Regressing**

**ATOC7500\_applicationlab2\_AR1\_regression\_AO.ipynb**

**LEARNING GOALS:**

1) Calculate and analyze the autocorrelation at a range of lags using output from an EOF analysis (the Arctic Oscillation Index).

2) Generate a red noise time series with equivalent memory as an observed time series (i.e., given lag-1 autocorrelation).

3) Correlate two time series and calculate the statistical significance.

4) Evaluate the statistical significance obtained in the context of the number of chances provided for success. What happens when you go “fishing” for correlations and give yourself lots of opportunity for success? Can you critically evaluate the chances that your regression is statistically different than 0 just by chance?

**DATA and UNDERLYING SCIENCE:**

In this notebook, you will analyze the monthly Arctic Oscillation (AO) timeseries from January 1950 to present. The AO timeseries comes from an Empirical Orthogonal Function (EOF) analysis. We will implement EOFs in the next application lab so in this lab we are actually using multiple analysis methods introduced in this class, some that you have learned and some that you are still yet to learn ☺.

How do you find the AO value each month? To identify the atmospheric circulation patterns that explain the most variance, NOAA regularly applies EOF analysis to the monthly mean 1000-hPa height anomalies poleward of 20° latitude for the Northern Hemisphere. The AO spatial pattern (Figure 1 below) emerges as the first EOF (explaining the most variance, 19%). The AO timeseries we will analyze is a measure of the amplitude of the pattern in Figure 1 in a given month. In other words – the AO timeseries is the first principal component (a timeseries) associated with the first EOF (a spatial structure). More information on the EOF analysis here:

http://www.cpc.ncep.noaa.gov/products/precip/CWlink/daily\_ao\_index/history/method.shtml



Figure 1. The loading pattern of the Arctic Oscillation (AO), i.e., the structure explaining the most variance of monthly mean 1000mb height during 1979-2000 period. In other words – this is the first EOF.

The data are available and regularly updated here:

<http://www.cpc.ncep.noaa.gov/products/precip/CWlink/pna/norm.nao.monthly.b5001.current.ascii>

You can work with the data directly on the web (assuming you have an internet connection). I have also downloaded the data and made them available – The name of the data file is “monthly.ao.index.b50.current.ascii”.

**Questions to guide your analysis of Notebook #2:**

1) Start with the default settings in the code. First read in the Arctic Oscillation (AO) data. Look at your data!! Plot it as a timeseries. Save the timeseries plot as a postscript file and put it in this document.

Here is the time series of the Arctic Oscillation data:



2) Calculate the lag-one autocorrelation (AR1) of the AO data and record it here. Use two methods (np.correlate, dot products). Check that they give you the same result. Interpret the value. How much memory (red noise) is there in the AO from month to month?

The lag-one correlation yields r = 0.31526 for np.correlate and the same value for the direct calculation. If we square the autocorrelation then we get r2 = 0.0994, which is indicates that the lagged time series explains about 9% of the variance in the AO time series.

3) Calculate and plot the autocorrelation of the AO data at all lags. Describe your results. How red are the data at lags other than lag=1? Is there any interesting behavior of the autocorrelation as a function of lag? What would you expect for red noise timeseries with an AR1=value reported in 2)?

The autocorrelation of the AO data at all lags shows there is some red noise in the time series, particularly in lags between -5 and 5. However, it also looks like there is some periodicity to the autocorrelation at a lag of +-6 and +-7. For a red noise time series with the AR1 value from (2), I would expect the percent variance explained to be 0.0181%, which is much lower than the variance explained calculated above.



4) Generate a synthetic red noise time series with the same lag-1 autocorrelation as the AO data. Your synthetic dataset should have different time evolution but the same memory as the AO. Plot the AO timeseries and the synthetic red noise time series. Put the plot below.

A picture containing lined, beach, sitting, standing

Description automatically generated

5) Do you expect to find any correlation between the two datasets, i.e., the synthetic red noise and the actual AO data? What is the correlation between the synthetic red noise and the actual AO data? Calculate a regression coefficient and other associated regression statistics.

The correlation between the synthetic red noise and the actual AO data is -0.01345, the slope is -0.01366, the p value is 0.696, and the standard error is 0.0349. The r squared value is 0.0001810, meaning the percent variance in the AO time series explained by the red noise is 0.0181%.

6) Next -- Have some fun and go “fishing for correlations”. What happens if you try correlating subsets of the two datasets many times? When you try 200 times -- what is the maximum correlation/variance explained you can obtain between the synthetic red noise and the actual data? *Note: you are effectively searching for a high correlation with no a priori reason to do so.... THIS IS NOT good practice for science but we are doing it here because it is instructive to see what happens :)*

When you try 200 times, the maximum variance is 173. The largest r value or correlation is 0.67, and the associated largest variance explained is 44.62%.

7) Calculate the correlation statistics for the highest correlation obtained in question 6). Two methods are provided - they should give you the same answers. Place a confidence interval on your correlation. Because you have found a correlation that is not equal to 0, use the Fisher-Z Transformation. Did your "fishing" for a statistically significant correlation work? Is your highest correlation statistically significant (i.e., can you reject the null hypothesis that the correlation is zero)? Write out the steps for hypothesis testing and use the values you calculate to formally assess.

For the highest correlation in (6), the r value in both methods is 0.668. The confidence interval is 0.29 to 0.87, which does not include zero and so we can reject the null hypothesis that the correlation is zero. So yes – the “fishing” for a statistically significant correlation worked.

8) You went searching for correlations, you searched long and hard (200 times!) You should have been concerned that the largest correlation you found would be a false positive. Do you think you found a false positive? Explain what you found and potentially why you think it is important statistically but not physically. What lessons did you learn by “fishing for correlations”?

I think that if you try enough correlations for your dataset, then you are likely to come up with some sort of statistically significant correlation. Fishing for correlations is unlikely to yield a true positive, and in this case, I think we did find a false positive. The probability of correctly rejecting all null hypotheses is 0.0035 %, which is extremely low.