# Lab 7 – Data Preprocessing II

**Submission:** `lab07.ipynb`, pushed to your Git repo on `master`.
**Points**: **20**
**Due**: Monday, February 11, 9:59am

## Objectives

- Munge this!
- Cleaning messy data by integrating other data
- Work on different techniques to assess similar variables

## Introduction

You already started the first part of this lab in Lab 6, in which you downloaded a year of hourly weather observations. If not, be sure to complete the first part of the lab before you begin.

## Preparing for your lab

Create a **`lab07.ipynb`** file. Create your header cell, then your first cell to set up your imports:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

Work through this lab, and enter the answers to questions that are scattered throughout this lab.

> As usual, pay attention to the [P] vs. [M] indicator. All [P] questions should be answered by writing the Python code that outputs the answer. Your code quality should be striving for smart, efficient, well-designed, well-documented code. For code that is asking for specific answers, you should be writing Python code that outputs the answer that is wanted, and nothing more.

In your last lab, you created a **`data`** directory, and downloaded a dataset from *The Pennsylvania State Climatologist* at http://climate.psu.edu/data/ida representing Williamsport, PA (KIPT). The last lab focused quite a bit on preliminary steps to clean the data. You also had your first real exposure to dealing with times and dates with data, and understood first-hand why it's so important to be sure you set the proper type of every variable in your data. You should have printed out summary statistics for every variable, which should have included the occurrence of missing (e.g. `NaN, null`) values.

There are two important observations to make here:

1) You had missing data
2) The data that you do have may be questionable, as it is quite noisy at times.

How will we create a complete dataset?

# Your task

The scenario for this lab is as follows. You have some missing data, and you've decided to estimate the data from nearby stations. This is a data integration challenge. KIPT is a pretty reliable station, but there are missing values.

# Exercises

1) [P] Create a Python function called `process_FAA_hourly_data` that takes a filename as a string, and returns a completely processed pandas data frame, ready for analysis. It should do everything that the previous lab did to clean the file, including converting all numeric variables to their simplest numeric types, and converting the date/time stamp (first variable) to a pandas `DatestampIndex`, which becomes the actual index for the data frame. It should drop the date time variable after moving it to become the index.

   You completed this functionality in the last lab. (The last lab also had you create a new categorical variable called `"Quarter"`. Do not include that functionality in this function.) Just copy over those important statements for processing FAA hourly files to cleaned data frames.

2) [P] Use your new function to read in the KIPT data file you downloaded in the last lab. Output the results of `info()` and `describe()` to confirm you read it in correctly.

3) [P] Read in the file **`FAA_PA_stations.csv`** provided on Moodle. It's not actually a comma separated file, but a *tab* separated file. Store the data frame as `stations`.

4) [P] Examine the data frame (show the first few records). In particular, pay close attention to the variables `Lat` and `Lon`. These represent the precise latitude and longitude geolocation for the weather station.

   Then, create a new variable in `stations` that stores the distance of every station in PA to Williamsport (KIPT). Use a standard Euclidean distance calculation (over latitude and longitude) to compute the distance between the stations. As a reminder, Euclidean distance between two points defined by $(x_1, y_1)$ and $(x_2, y_2)$ is:

$$dist = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

5) [P] Output the top 5 stations that are closest to KIPT. (The closest one should be to itself!) The stations should be listed in order of increasing distance from KIPT.

6) [P] Using your results, go back to the PSU climate website, and download the faa_hourly data for the same date ranges for the **three** closest stations to KIPT. Copy them into your data folder. Then, read in each data file into its own data frame using your function. You should have four data frames: `df_kipt`, and three other data frames representing the three closest stations.

7) [P] Create a function called `get_missing_timestamps` that takes a data frame of FAA hourly data, and returns a simple Python list of `Timestamp` objects representing all of the dates that are missing. (You completed some of this functionality in the previous lab. Now you are making it a useful function.) The design of the function is up to you. However, a forward thinking data scientist will make these functions as flexible as possible. (AND, a good software developer will NOT hardcode values for specific cases!)

8) [P] Write the Python code that reports a list of timestamps that are included in each nearby station that KIPT is missing. (HINT: it will be far, far easier if you consider casting your `list` to a `set`, and using `set` methods.)

Now, your real task. Some instructor of yours (not mentioning any names) loves *complete* datasets. He gets freaked out when he sees missing data. He's also not a fan of noise. Noise be bad!

You're going to repair this data by filling in the best representative data from a nearby station. For the remainder of this exercise, we will focus on the **average temperature** attribute (column 1). And, at this point, you should be not only going for the most efficient code, but you are going to start doing more advanced analyses. So, when your output is not obvious or intuitive, please **write markdown after your output is generated that explains your code, and interprets your results!**

9) [P] Since we're going to focus on average temperature, create a new data frame called `df_ave_temps` that contains a the average temperature from all four stations. It should have a COMPLETE hourly date range for its index from the specified start date to finish date (i.e. you should have 365 * 24 rows.) If the dates are missing from the station you are copying from, then plug in a NaN value for that entry. Label your variables accordingly. You will use these data for the remainder of this work.

10) [P] Report the dates that have missing data for all four stations, if any.

11) [P] Start by creating three separate scatter plots showing the average temperature over time for Williamsport and each of the three closest stations you downloaded. Plot KIPT with the same color for each plot, but use a different color for each of the close stations. Be sure to label your plots accordingly.

12) [P] The lines will largely overlap, as one would expect. Select one month of your choosing, and repeat the previous step for that month.

13) [P] Report the number of NaN values that are shared between KIPT and each station you downloaded for average temperature. Remember, be careful how you compare values. Dates will not be in the same row index due to missing data (i.e. do not just iterate over rows using integers!) Write the code to generate the answer. Because you are focused on cleaning up average temperature, this could help you decide which station to use.

14) [P] Perhaps it's more important to select the station that has the most similar values. Write a function called `compare_station` that takes two `Series` objects, and computes the sum of the absolute value of the difference between each number in the vector. You should only sum the values that have valid values for both entries. Return the average of these absolute differences. Then, call `compare_station` on KIPT and each of the new station, but pass only the average temp vector from each station.

15) [P] As we learned in class, you could compute a *correlation coefficient* between columns of data to determine similarity. Compute the correlation coefficient between the average temp of KIPT, and each of the other stations you downloaded. They should all be very close to 1, but not quite. What does this technique suggest which station is most similar?

16) [M] Important question: Which station seems to be the best representative for KIPT? Why?

17) [P] Create a new attribute called `aveTempFixed` in your KIPT data frame that keeps all of the original average temp data, but takes the readings from the closest station to replace in the NA values.

18) [P] How many missing values did you fill in?

19) [P] Create a new function called `moving_ave` that takes a Series of numbers and a window size, and returns a new vector that contains the moving average over the window size. For example:

```
moving_ave(pd.Series([5,10,3,7,8,9]),3)
```

```
NA, 6, 6.666667, 6, 8, NA
```

NOTE: Your function should center your data over the range, and fill the ends with NA values.

20) Use your new function to smooth out your fixed average temp data. Call moving_ave with a moving average window size of 6, 12, and 24. Show the raw data and your three smoothed plots on one single plot. Interpret your results.

## Deliverables

**Commit and push `lab07.ipynb`. Be sure you have every cell run, and output generated. All plots should have plt.show() as the last command to ensure the plot is generated and viewable on Gitlab. Verify that your file is pushed properly on Gitlab. Be sure each question is properly annotated.**