SIG731 2023: Task 4P

Working with **pandas** Data Frames (Heterogeneous Data)

Last updated: 2023-11-24

Contents

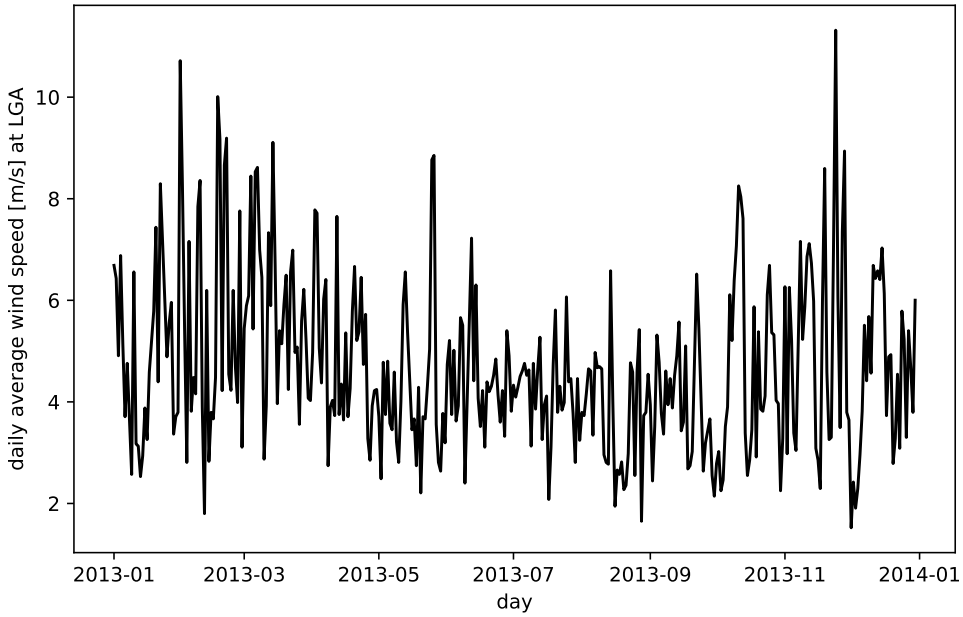
1 Task

Download the `nycflights13_weather.csv.gz` data file from our unit site (*Learning Resources → Data*). It gives the hourly meteorological data for three airports in New York: LGA, JFK, and EWR for the whole year of 2013. The columns are:

- `origin` – weather station: LGA, JFK, or EWR,
- `year`, `month`, `day`, `hour` – time of recording,
- `temp`, `dewp` – temperature and dew point in degrees Fahrenheit,
- `humid` – relative humidity,
- `wind_dir`, `wind_speed`, `wind_gust` – wind direction (in degrees), speed and gust speed (in mph),
- `precip` – precipitation, in inches,
- `pressure` – sea level pressure in millibars,
- `visib` – visibility in miles,
- `time_hour` – date and hour (based on the `year`, `month`, `day`, `hour` fields) formatted as `YYYY-mm-dd HH:MM:SS` (actually, `YYYY-mm-dd HH:00:00`). *However, due to a bug in the dataset, the data in this column are (incorrectly!) shifted by 1 hour. Do not rely on it unless you manually correct it.*

Then, create a single Jupyter/IPython notebook (see the *Artefacts* section below for all the requirements), where you perform what follows.

1. Convert all columns so that they use metric (International System of Units, SI) or derived units: `temp` and `dewp` to Celsius, `precip` to millimetres, `visib` to metres, as well as `wind_speed` and `wind_gust` to metres per second. Replace the data in-place (overwrite existing columns with new ones).
2. Compute *daily* mean wind speeds for the `LGA` airport (~365 total speed values, for each day separately; you can, for example, group the data by year, month, and day at the same time).
3. Present the daily mean wind speeds at `LGA` (~365 aforementioned data points) in a single plot, e.g., using the `matplotlib.pyplot.plot` function. The x-axis labels should be human-readable and intuitive (e.g., month names or dates). Reference result:



4. Identify the ten windiest days at `LGA` (dates and the corresponding mean *daily* wind speeds).

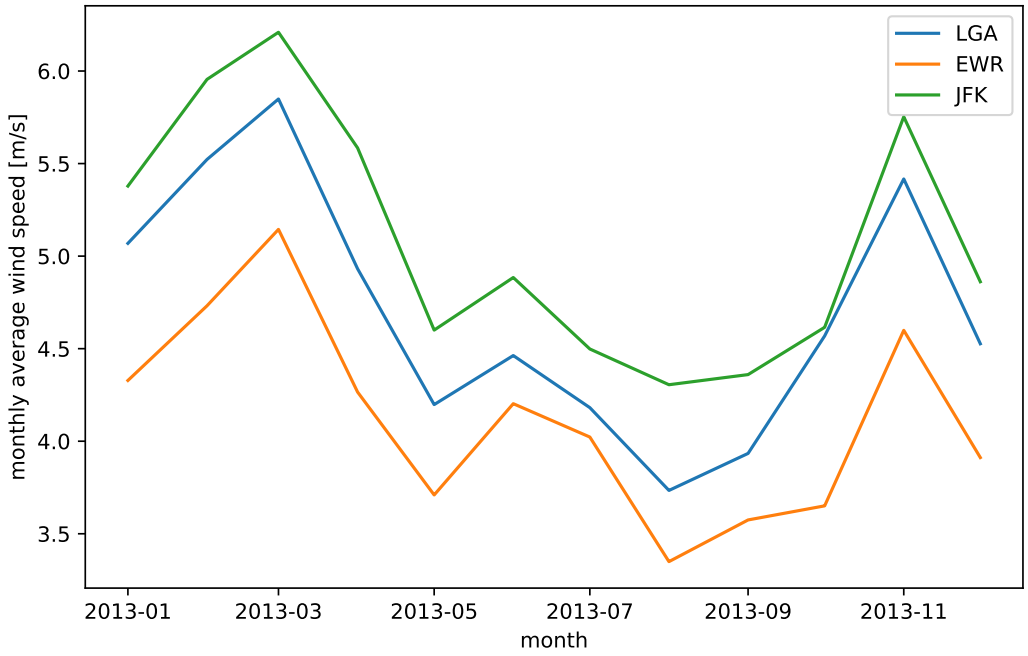
   Reference result:

```
##             wind_speed
## date
## 2013-11-24   11.317783
## 2013-01-31   10.717598
## 2013-02-17   10.010236
## 2013-02-21    9.192903
## 2013-02-18    9.174264
## 2013-03-14    9.109958
## 2013-11-28    8.938477
## 2013-05-26    8.852736
## 2013-05-25    8.766995
## 2013-02-20    8.659819
```

5. Compute the monthly mean wind speeds for all the three airports.

   There is one obvious outlier amongst the observed wind speeds. Locate it (programmatically, do not hardcode the date/day/row number) and replace it with `np.nan` (NaN) before computing the means.

6. Draw the monthly mean wind speeds for the three airports on the same plot (three curves of different colours). Add a readable legend. Reference result:



2 Artefacts

Make sure that your notebook has a **readable structure**; in particular, that it is divided into sections. Use rich Markdown formatting (text in dedicated Markdown chunks – not just Python comments).

Do not include the questions/tasks from the task specification. Your notebook should read nicely and smoothly – like a report from data analysis that you designed yourself. Make the flow read natural (e.g., *First, let us load the data on… Then, let us determine… etc.*). Imagine it is a piece of work that you would like to show to your manager or clients — you certainly want to make a good impression. Check your spelling and grammar. Also, use formal language.

At the start of the notebook, you need to provide: the **title** of the report (e.g., *Task 42: How Much I Love This Unit*), your **name**, **student number**, and **email address**.

Then, add 1–2 introductory paragraphs (an introduction/abstract – what the task is about).

Before each nontrivial code chunk, briefly **explain** what its purpose is. After each code chunk, **summarise and discuss the obtained results** (in a few sentences).

Conclude the report with 1–2 paragraphs (summary/discussion/possible extensions of the analysis etc.).

Checklist:

1. Header, introduction, conclusion (Markdown chunks).
2. Text divided into sections, all major code chunks commented and discussed in your own words (Markdown chunks).
3. Every subtask addressed/solved. In particular, all reference results that are part of the task specification have been reproduced (plots, computed aggregates, etc.).
4. The report is readable and neat. In particular:

- all code lines are visible in their entirety (they are not too long),
- code chunks use consecutive numbering (select *Kernel - Restart and Run All* from the Jupyter menu),
- rich Markdown formatting is used (`# Section Title`, `* bullet list`, `1. enumerated list`, `| table |`, `*italic*`, etc.),
- the printing of unnecessary/intermediate objects is minimised (focus on reporting the results specifically requested in the task specification).

Submissions which do not *fully* (100%) conform to the task specification *on* the cut-off date will be marked as FAIL.

Good luck!