Wind Statistics

Introduction:

The data have been modified to contain some missing values, identified by NaN.
Using pandas should make this exercise easier, in particular for the bonus question.

You should be able to perform all of these operations without using a for loop or other looping construct.

1. The data in 'wind.data' has the following format:

In [1]:

```
"""
Yr Mo Dy   RPT   VAL   ROS   KIL   SHA   BIR   DUB   CLA   MUL   CLO   BEL   MAL
61  1  1 15.04 14.96 13.17  9.29   NaN  9.87 13.67 10.25 10.83 12.58 18.50 15.04
61  1  2 14.71   NaN 10.83  6.50 12.62  7.67 11.50 10.04  9.79  9.67 17.54 13.83
61  1  3 18.50 16.88 12.33 10.13 11.17  6.17 11.25   NaN  8.50  7.67 12.75 12.71
"""
```

Out[1]:

```
'\nYr Mo Dy   RPT   VAL   ROS   KIL   SHA   BIR   DUB   CLA   MUL   CLO   BEL   MAL\n61  1  1 15.04 14.96 13.17  9.29   NaN  9.87 13.67 10.25 10.83 12.58 18.5
0 15.04\n61  1  2 14.71   NaN 10.83  6.50 12.62  7.67 11.50 10.04  9.79  9.67 17.54 13.83\n61  1  3 18.50 16.88 12.33 10.13 11.17  6.17 11.25   NaN  8.50  7.6
7 12.75 12.71\n'
```

The first three columns are year, month and day. The remaining 12 columns are average windspeeds in knots at 12 locations in Ireland on that day.

More information about the dataset go here.

Step 1. Import the necessary libraries

In [2]:

```python
import pandas as pd
import datetime
```

Step 2. Import the dataset from this address

Step 3. Assign it to a variable called data and replace the first 3 columns by a proper datetime index.

In [3]:

```python
# parse_dates gets 0, 1, 2 columns and parses them as the index
data_url = 'https://raw.githubusercontent.com/guipsamora/pandas_exercises/master/06_Stats/Wind_Stats/wind.data'
data = pd.read_csv(data_url, sep = "\s+", parse_dates = [[0,1,2]])
data.head()
```

Out[3]:

	Yr_Mo_Dy	RPT	VAL	ROS	KIL	SHA	BIR	DUB	CLA	MUL	CLO	BEL	MAL
0	2061-01-01	15.04	14.96	13.17	9.29	NaN	9.87	13.67	10.25	10.83	12.58	18.50	15.04
1	2061-01-02	14.71	NaN	10.83	6.50	12.62	7.67	11.50	10.04	9.79	9.67	17.54	13.83
2	2061-01-03	18.50	16.88	12.33	10.13	11.17	6.17	11.25	NaN	8.50	7.67	12.75	12.71
3	2061-01-04	10.58	6.63	11.75	4.58	4.54	2.88	8.63	1.79	5.83	5.88	5.46	10.88
4	2061-01-05	13.33	13.25	11.42	6.17	10.71	8.21	11.92	6.54	10.92	10.34	12.92	11.83

Step 4. Year 2061? Do we really have data from this year? Create a function to fix it and apply it.

In [4]:

```python
data["Yr_Mo_Dy"].dtypes
```

Out[4]:

```
dtype('<M8[ns]')
```

In [5]:

```python
def fix_century(x):
    year = x.year-100 if x.year>1989 else x.year
    return datetime.date(year, x.month, x.day)
```

In [6]:

```python
data['Yr_Mo_Dy'] = data['Yr_Mo_Dy'].apply(fix_century)
data.head()
```

Out[6]:

	Yr_Mo_Dy	RPT	VAL	ROS	KIL	SHA	BIR	DUB	CLA	MUL	CLO	BEL	MAL
0	1961-01-01	15.04	14.96	13.17	9.29	NaN	9.87	13.67	10.25	10.83	12.58	18.50	15.04
1	1961-01-02	14.71	NaN	10.83	6.50	12.62	7.67	11.50	10.04	9.79	9.67	17.54	13.83
2	1961-01-03	18.50	16.88	12.33	10.13	11.17	6.17	11.25	NaN	8.50	7.67	12.75	12.71
3	1961-01-04	10.58	6.63	11.75	4.58	4.54	2.88	8.63	1.79	5.83	5.88	5.46	10.88
4	1961-01-05	13.33	13.25	11.42	6.17	10.71	8.21	11.92	6.54	10.92	10.34	12.92	11.83

Step 5. Set the right dates as the index. Pay attention at the data type, it should be datetime64[ns].

In [7]:

```python
data["Yr_Mo_Dy"] = pd.to_datetime(data["Yr_Mo_Dy"])
data = data.set_index('Yr_Mo_Dy')
data.head()
```

Out[7]:

Yr_Mo_Dy	RPT	VAL	ROS	KIL	SHA	BIR	DUB	CLA	MUL	CLO	BEL	MAL
1961-01-01	15.04	14.96	13.17	9.29	NaN	9.87	13.67	10.25	10.83	12.58	18.50	15.04
1961-01-02	14.71	NaN	10.83	6.50	12.62	7.67	11.50	10.04	9.79	9.67	17.54	13.83
1961-01-03	18.50	16.88	12.33	10.13	11.17	6.17	11.25	NaN	8.50	7.67	12.75	12.71
1961-01-04	10.58	6.63	11.75	4.58	4.54	2.88	8.63	1.79	5.83	5.88	5.46	10.88
1961-01-05	13.33	13.25	11.42	6.17	10.71	8.21	11.92	6.54	10.92	10.34	12.92	11.83

Step 6. Compute how many values are missing for each location over the entire record.

They should be ignored in all calculations below.

In [8]:

```python
data.isna().sum()
```

Out[8]:

```
RPT    6
VAL    3
ROS    2
KIL    5
SHA    2
BIR    0
DUB    3
CLA    2
MUL    3
CLO    1
BEL    0
MAL    4
dtype: int64
```

Step 7. Compute how many non-missing values there are in total.

In [9]:

```python
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 6574 entries, 1961-01-01 to 1978-12-31
Data columns (total 12 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   RPT     6568 non-null   float64
 1   VAL     6571 non-null   float64
 2   ROS     6572 non-null   float64
 3   KIL     6569 non-null   float64
 4   SHA     6572 non-null   float64
 5   BIR     6574 non-null   float64
 6   DUB     6571 non-null   float64
 7   CLA     6572 non-null   float64
 8   MUL     6571 non-null   float64
 9   CLO     6573 non-null   float64
 10  BEL     6574 non-null   float64
 11  MAL     6570 non-null   float64
dtypes: float64(12)
memory usage: 667.7 KB
```

In [10]:

```python
data.notnull().sum()
```

Out[10]:

```
RPT    6568
VAL    6571
ROS    6572
KIL    6569
SHA    6572
BIR    6574
DUB    6571
CLA    6572
MUL    6571
CLO    6573
BEL    6574
MAL    6570
dtype: int64
```

Step 8. Calculate the mean windspeeds of the windspeeds over all the locations and all the times.

A single number for the entire dataset.

In [11]:

```python
data.sum().sum() / data.notna().sum().sum()
```

Out[11]:

```
10.227883764282181
```

Step 9. Create a DataFrame called loc_stats and calculate the min, max and mean windspeeds and standard deviations of the windspeeds at each location over all the days

A different set of numbers for each location.

In [12]:

```python
data.describe()
```

Out[12]:

	RPT	VAL	ROS	KIL	SHA	BIR	DUB	CLA	MUL	CLO	BEL	MAL
count	6568.000000	6571.000000	6572.000000	6569.000000	6572.000000	6574.000000	6571.000000	6572.000000	6571.000000	6573.000000	6574.000000	6570.000000
mean	12.362987	10.644314	11.660526	6.306468	10.455834	7.092254	9.797343	8.495053	8.493590	8.707332	13.121007	15.599079
std	5.618413	5.267356	5.008450	3.605811	4.936125	3.968683	4.977555	4.499449	4.166872	4.503954	5.835037	6.699794
min	0.670000	0.210000	1.500000	0.000000	0.130000	0.000000	0.000000	0.000000	0.000000	0.040000	0.130000	0.670000
25%	8.120000	6.670000	8.000000	3.580000	6.750000	4.000000	6.000000	5.090000	5.370000	5.330000	8.710000	10.710000
50%	11.710000	10.170000	10.920000	5.750000	9.960000	6.830000	9.210000	8.080000	8.170000	8.290000	12.500000	15.000000
75%	15.920000	14.040000	14.670000	8.420000	13.540000	9.670000	12.960000	11.420000	11.190000	11.630000	16.880000	19.830000
max	35.800000	33.370000	33.840000	28.460000	37.540000	26.160000	30.370000	31.080000	25.880000	28.210000	42.380000	42.540000

Step 10. Create a DataFrame called day_stats and calculate the min, max and mean windspeed and standard deviations of the windspeeds across all the locations at each day.

A different set of numbers for each day.

In [13]:

```python
day_stats = pd.DataFrame()

day_stats['min'] = data.min(axis = 1) # min
day_stats['max'] = data.max(axis = 1) # max
day_stats['mean'] = data.mean(axis = 1) # mean
day_stats['std'] = data.std(axis = 1) # standard deviations

day_stats.head()
```

Out[13]:

Yr_Mo_Dy	min	max	mean	std
1961-01-01	9.29	18.50	13.018182	2.808875
1961-01-02	6.50	17.54	11.336364	3.188994
1961-01-03	6.17	18.50	11.641818	3.681912
1961-01-04	1.79	11.75	6.619167	3.198126
1961-01-05	6.17	13.33	10.630000	2.445356

Step 11. Find the average windspeed in January for each location.

Treat January 1961 and January 1962 both as January.

In [14]:

```python
data.loc[data.index.month==1].mean()
```

Out[14]:

```
RPT    14.847325
VAL    12.914560
ROS    13.299624
KIL     7.199498
SHA    11.667734
BIR     8.054839
DUB    11.819355
CLA     9.512047
MUL     9.543208
CLO    10.053566
BEL    14.550520
MAL    18.028763
dtype: float64
```

Step 12. Downsample the record to a yearly frequency for each location.

In [15]:

```python
data.groupby(data.index.to_period('A')).mean()
```

Out[15]:

Yr_Mo_Dy	RPT	VAL	ROS	KIL	SHA	BIR	DUB	CLA	MUL	CLO	BEL	MAL
1961	12.299583	10.351796	11.362369	6.958227	10.881763	7.729726	9.733923	8.858788	8.647652	9.835577	13.502795	13.680773
1962	12.246923	10.110438	11.732712	6.960440	10.657918	7.393068	11.020712	8.793753	8.316822	9.676247	12.930685	14.323956
1963	12.813452	10.836986	12.541151	7.330055	11.724110	8.434712	11.075699	10.336548	8.903589	10.224438	13.638877	14.999014
1964	12.363661	10.920164	12.104372	6.787787	11.454481	7.570874	10.259153	9.467350	7.789016	10.207951	13.740546	14.910301
1965	12.451370	11.075534	11.848767	6.858466	11.024795	7.478110	10.618712	8.879918	7.907425	9.918082	12.964247	15.591644
1966	13.461973	11.557205	12.020630	7.345726	11.805041	7.793671	10.579808	8.835096	8.514438	9.768959	14.265836	16.307260
1967	12.737151	10.990986	11.739397	7.143425	11.630740	7.368164	10.652027	9.325616	8.645014	9.547425	14.774548	17.135945
1968	11.835628	10.468197	11.409754	6.477678	10.760765	6.067322	8.859180	8.255519	7.224945	7.832978	12.808634	15.017486
1969	11.166356	9.723699	10.902000	5.767973	9.873918	6.189973	8.564493	7.711397	7.924521	7.754384	12.621233	15.762904
1970	12.600329	10.726932	11.730247	6.217178	10.567370	7.609452	9.609890	8.334630	9.297616	8.289808	13.183644	16.456027
1971	11.273123	9.095178	11.088329	5.241507	9.440329	6.097151	8.385890	6.757315	7.915370	7.229753	12.208932	15.025233
1972	12.463962	10.561311	12.058333	5.929699	9.430410	6.358825	9.704508	7.680792	8.357295	7.515273	12.727377	15.028716
1973	11.828466	10.680493	10.680493	5.547863	9.640877	6.548740	8.482110	7.614274	8.245534	7.812411	12.169699	15.441096
1974	13.643096	11.811781	12.336356	6.427041	11.110986	6.809781	10.084603	9.896986	9.331753	8.736356	13.252959	16.947671
1975	12.008575	10.293836	11.564712	5.269096	9.190082	5.668521	8.562603	7.843836	8.797945	7.382822	12.631671	15.307863
1976	11.737842	10.203115	10.761230	5.109426	8.846339	6.311038	9.149126	7.146202	8.883716	7.883087	12.332377	15.471448
1977	13.099616	11.144493	12.627836	6.073945	10.003836	8.586438	11.523205	8.378384	9.098192	8.821616	13.459068	16.590849
1978	12.504356	11.044274	11.380000	6.082356	10.167233	7.650658	9.489342	8.800466	9.089753	8.301699	12.967397	16.771370

Step 13. Downsample the record to a monthly frequency for each location.

In [16]:

```python
data.groupby(data.index.to_period('M')).mean()
```

Out[16]:

Yr_Mo_Dy	RPT	VAL	ROS	KIL	SHA	BIR	DUB	CLA	MUL	CLO	BEL	MAL
1961-01	14.841333	11.988333	13.431613	7.736774	11.072759	8.588065	11.184839	9.245333	9.085806	10.107419	13.880968	14.703226
1961-02	16.269286	14.975357	14.441481	9.230741	13.852143	10.937500	11.890714	11.846071	11.821429	12.714286	18.583214	15.411786
1961-03	10.890000	11.296452	10.752903	7.284000	10.509355	8.866774	9.644194	9.829677	10.294138	11.251935	16.410968	15.720000
1961-04	10.722667	9.427667	9.998000	5.830667	8.435000	6.495000	6.925333	7.094667	7.342333	7.237000	11.147333	10.278333
1961-05	9.860968	8.850000	10.818065	5.905333	9.490323	6.574839	7.604000	8.177097	8.039355	8.499355	11.900323	12.011613
...	...	...	...	...	...	...	...	...	...	...	...	...
1978-08	9.645161	8.259355	9.032258	4.502903	7.368065	5.935161	5.650323	5.417742	7.241290	5.536774	10.466774	12.054194
1978-09	10.913667	10.895000	10.635000	5.725000	10.372000	9.278333	10.790333	9.583000	10.069333	8.939000	15.680333	19.391333
1978-10	9.897742	8.670968	9.295806	4.721290	8.525161	6.774194	8.115484	7.337742	8.297742	8.243871	13.776774	17.150000
1978-11	16.151667	14.802667	13.508000	7.317333	11.475000	8.743000	11.492333	9.657333	10.701333	10.676000	17.404667	20.723000
1978-12	16.175484	13.748065	15.635161	7.094839	11.398710	9.241613	12.077419	10.194839	10.616774	11.028710	13.859677	21.371613

216 rows × 12 columns

Step 14. Downsample the record to a weekly frequency for each location.

In [17]:

```python
data.groupby(data.index.to_period('W')).mean()
```

Out[17]:

Yr_Mo_Dy	RPT	VAL	ROS	KIL	SHA	BIR	DUB	CLA	MUL	CLO	BEL	MAL
1960-12-26/1961-01-01	15.040000	14.960000	13.170000	9.290000	NaN	9.870000	13.670000	10.250000	10.830000	12.580000	18.500000	15.040000
1961-01-02/1961-01-08	13.541429	11.486667	10.487143	6.417143	9.474286	6.435714	11.061429	6.616667	8.434286	8.497143	12.481429	13.238571
1961-01-09/1961-01-15	12.468571	8.967143	11.958571	4.630000	7.351429	5.072857	7.535714	6.820000	5.712857	7.571429	11.125714	11.024286
1961-01-16/1961-01-22	13.204286	9.862857	12.982857	6.328571	8.966667	7.417143	9.257143	7.875714	7.145714	8.124286	9.821429	11.434286
1961-01-23/1961-01-29	19.880000	16.141429	18.225714	12.720000	17.432857	14.828571	15.528571	15.160000	14.480000	15.640000	20.930000	22.530000
...	...	...	...	...	...	...	...	...	...	...	...	...
1978-11-27/1978-12-03	14.934286	11.232857	13.941429	5.565714	10.215714	8.618571	9.642857	7.685714	9.011429	9.547143	11.835714	18.728571
1978-12-04/1978-12-10	20.740000	19.190000	17.034286	9.777143	15.287143	12.774286	14.437143	12.488571	13.870000	14.082857	18.517143	23.061429
1978-12-11/1978-12-17	16.758571	14.692857	14.987143	6.917143	11.397143	7.272857	10.208571	7.967143	9.168571	8.565714	11.102857	15.562857
1978-12-18/1978-12-24	11.155714	8.008571	13.172857	4.004286	7.825714	6.290000	7.798571	8.667143	7.151429	8.072857	11.845714	18.977143
1978-12-25/1978-12-31	14.951429	11.801429	16.035714	6.507143	9.660000	8.620000	13.708571	10.477143	10.868571	11.471429	12.947143	26.844286

940 rows × 12 columns

Step 15. Calculate the min, max and mean windspeeds and standard deviations of the windspeeds across all locations for each week (assume that the first week starts on January 2 1961) for the first 52 weeks.

In [18]:

```python
weekly = data.resample('W').agg(['min','max','mean','std'])
weekly.loc[weekly.index[1:53], "RPT":"MAL"] .head(10)
```

Out[18]:

	RPT				VAL				ROS		...	CLO		BEL				MAL			
	min	max	mean	std	min	max	mean	std	min	max	...	mean	std	min	max	mean	std	min	max	mean	std
Yr_Mo_Dy																					
1961-01-08	10.58	18.50	13.541429	2.631321	6.63	16.88	11.486667	3.949525	7.62	12.33	...	8.497143	1.704941	5.46	17.54	12.481429	4.349139	10.88	16.46	13.238571	1.773062
1961-01-15	9.04	19.75	12.468571	3.555392	3.54	12.08	8.967143	3.148945	7.08	19.50	...	7.571429	4.084293	5.25	20.71	11.125714	5.552215	5.17	16.92	11.024286	4.692355
1961-01-22	4.92	19.83	13.204286	5.337402	3.42	14.37	9.862857	3.837785	7.29	20.79	...	8.124286	4.783952	6.50	15.92	9.821429	3.626584	6.79	17.96	11.434286	4.237239
1961-01-29	13.62	25.04	19.880000	4.619061	9.96	23.91	16.141429	5.170224	12.67	25.84	...	15.640000	3.713368	14.04	27.71	20.930000	5.210726	17.50	27.63	22.530000	3.874721
1961-02-05	10.58	24.21	16.827143	5.251408	9.46	24.21	15.460000	5.187395	9.04	19.70	...	9.460000	2.839501	9.17	19.33	14.012857	4.210858	7.17	19.25	11.935714	4.336104
1961-02-12	16.00	24.54	19.684286	3.587677	11.54	21.42	16.417143	3.608373	13.67	21.34	...	14.440000	1.746749	15.21	26.38	21.832857	4.063753	17.04	21.84	19.155714	1.828705
1961-02-19	6.04	22.50	15.130000	5.064609	11.63	20.17	15.091429	3.575012	6.13	19.41	...	13.542857	2.531361	14.09	29.63	21.167143	5.910938	10.96	22.58	16.584286	4.685377
1961-02-26	7.79	25.80	15.221429	7.020716	7.08	21.50	13.625714	5.147348	6.08	22.42	...	12.730000	4.920064	9.59	23.21	16.304286	5.091162	6.67	23.87	14.322857	6.182283
1961-03-05	10.96	13.33	12.101429	0.997721	8.83	17.00	12.951429	2.851955	8.17	13.67	...	12.370000	1.593685	11.58	23.45	17.842857	4.332331	8.83	17.54	13.951667	3.021387
1961-03-12	4.88	14.79	9.376667	3.732263	8.08	16.96	11.578571	3.230167	7.54	16.38	...	10.458571	3.655113	10.21	22.71	16.701429	4.358759	5.54	22.54	14.420000	5.769890

10 rows × 48 columns