PRINCIPLES AND METHODS OF DATA ANALYSIS IN THE ENERGY INDUSTRY

Laboratory Work No. 1

Reading, Cleaning, and Descriptive Analysis of Solar-Irradiance Time-Series Data

Related lecture: Lecture 1 — Exploratory Data Analysis & Python Foundations

Dataset: GECAD Photovoltaic (PV) system, January 2013, 5-minute sampling interval (file `PV2013January.xlsx`).

Purpose of the work

To gain practical experience working with a real-world sensor dataset supplied as a multi-sheet spreadsheet: extracting a single day of measurements, loading it into Python, and using basic language features — variables, conditional filtering, loops, and built-in aggregate functions — to compute descriptive statistics of a time series and to detect a meaningful sub-interval (the sunlit part of the day) from a threshold condition. The exercise also introduces a first, hands-on encounter with a common real-world problem: distinguishing genuine signal from sensor noise before trusting a computed result.

Background

1. About the dataset

The workbook contains one **Info** sheet documenting the measurement site and every column, plus one sheet per calendar day (e.g. `15-01-2013`), each holding 288 rows — one row every 5 minutes, for 24 hours. The columns relevant to this lab are:

Column	Meaning
A — TimeStamp	Time of the reading, hh:mm
B — ExlSolIrr	External radiation sensor, W/m^2 (documented in the Info sheet as "without data" — check before using any column blindly)
C — IntSolIrr	Integrated radiation sensor, W/m^2 — the column used in this lab

A first data-quality lesson: always read the Info/metadata sheet before trusting a column. In this dataset, column B (ExlSolIrr) is documented as having no data — and indeed every value in it is 0. Column C (IntSolIrr) is the one that actually carries the solar-radiation signal, which is why the task specifies it explicitly.

2. From spreadsheet to Python

The simplest path — and the one used in this lab — is to isolate one day's sheet and re-save it as a `.csv` file (Excel: right-click the sheet tab → Move or Copy → Create a copy → save the new workbook as CSV), then read it with pandas:

```
import pandas as pd

df = pd.read_csv('data_pv_15-01-2013.csv')
irr = df['IntSolIrr']          # column C as a pandas Series
```

It is also possible to read the original .xlsx directly with pandas (pd.read_excel(file, sheet_name='15-01-2013', header=2)), without a manual CSV export — worth trying as an alternative once the CSV-based version works.

3. Conditions and filtering a time series

A Boolean condition such as `irr > 0` applied to a whole pandas Series produces a Series of `True`/`False` values — one per row — which can then be used to select only the matching rows: `df[irr > 0]`. This is the vectorized equivalent of writing an `if` check inside a `for` loop over every row, and is the standard, preferred style in pandas.

> **Common pitfall: choosing the threshold**
>
> A plain `irr > 0` filter looks correct but can be misleading: many real sensors report small positive values from electrical noise even when there is no actual signal. Always inspect a few rows around the boundary before trusting a threshold — see the worked example below, where this exact issue appears in the data.

4. Aggregate functions vs. an explicit loop

pandas Series provide built-in aggregate methods — `.max()`, `.mean()`, `.idxmax()` (the index/position of the maximum) — that replace a hand-written accumulator loop. Both approaches are shown below for the maximum; the loop version is closer to what Lecture 1 covered, the vectorized version is what real code should use.

```
# Loop-based (Lecture 1 style)
max_val = irr[0]
for x in irr:
    if x > max_val:
        max_val = x

# Vectorized (idiomatic pandas)
max_val = irr.max()
```

5. A quick plot

Matplotlib's `plt.plot(x, y)` (Lecture 2) is enough to visualize a day's irradiance curve, and `ax.axvspan(start, end, ...)` highlights the detected daylight interval directly on the chart — see the worked example.

Task

Complete the following steps. Your variant (which calendar day to use) is assigned in the table further below.

- Download the data file PV2013January.xlsx.
- Open the file and locate the sheet for your assigned date (DD-01-2013). Copy that single sheet into a new, separate workbook.
- Save the new workbook in CSV format (e.g. data_pv_DD-01-2013.csv).
- Using the data in column C (IntSolIrr), write a program (Python or another language of your choice) that computes:
  - 4.1 — the maximum solar irradiance for the picked date;
  - 4.2 — the average solar irradiance for the picked date (over the full 24 hours);
  - 4.3 — the earliest time the sensor started receiving solar radiation;
  - 4.4 — the latest time the sensor received solar radiation;
  - 4.5 — the duration the sensor was receiving solar radiation, in both hours and as a percentage of 24 hours;
  - 4.6 — the average solar irradiance during the time the sensor was receiving radiation.

Save your code and take screenshots of the results where useful. Submit a report containing your code, screenshots/output, and results.

Optional / bonus tasks

If you want to go further, any of the following are good extensions and will be counted favorably:

- Compare the daylight window detected with a plain IntSolIrr > 0 filter against one using a small positive threshold (e.g. > 1 W/m^2). Explain, using your own data, why the two can disagree.
- Compute the total daily insolation received per square meter (in Wh/m^2), by summing IntSolIrr over all 5-minute samples and multiplying by 5/60. Compare your value across two or three different days.
- Produce a line chart of IntSolIrr vs. time for your date, with the detected daylight interval shaded (see the worked example below for a template).
- Implement steps 4.1-4.6 twice — once with an explicit loop, once with pandas' vectorized methods — and compare the length and readability of the two versions.
- Repeat the full analysis for a second date of your choice and compare the two days (e.g. a clear day vs. a cloudy day) in one short paragraph.

Variants

Your variant number is your position in the group list. Use the corresponding date from the table below (not every day of January is present in the file — missing days are weekends or days with no recorded data).

Variant	Date
1	02-01-2013
2	03-01-2013
3	04-01-2013

Variant	Date
4	08-01-2013
5	09-01-2013
6	10-01-2013
7	11-01-2013
8	12-01-2013
9	13-01-2013
10	14-01-2013
11	15-01-2013
12	16-01-2013
13	17-01-2013
14	18-01-2013
15	21-01-2013
16	22-01-2013
17	23-01-2013
18	24-01-2013
19	25-01-2013
20	26-01-2013
21	27-01-2013
22	28-01-2013
23	29-01-2013
24	30-01-2013
25	31-01-2013

If your group has more than 25 students, wrap around: variant 26 uses the same date as variant 1, variant 27 the same as variant 2, and so on.

Report requirements

Your submitted report must contain:

- the topic and purpose of the work;
- your variant number and assigned date;
- your program code (for all of steps 4.1-4.6, and for any bonus tasks attempted);

- the results of running your code (console output and/or screenshots), including numeric values for all six required quantities;
- a short conclusion: what the numbers show about your day, and anything unusual you noticed in the data (e.g. noise, missing values, cloud cover visible in the curve).

Worked example (teacher's variant)

Date used: **15-01-2013** (the same date used as an example in the original assignment). The full pipeline — from the extracted CSV to the final chart — is shown below.

Extracted data (first rows of data_pv_15-01-2013.csv)

```
TimeStamp,ExlSolIrr,IntSolIrr,OpTm,TmpAmb C,TmpMdul C,...
00:00,0,0,11123.82,-273.17,16.73,...
00:05,0,0.03,11123.9,-273.17,16.78,...
00:10,0,0,11123.99,-273.17,16.85,...
...
08:10,0,1.78,11131.98,-273.17,16.23,...
08:15,0,3.7,11132.06,-273.17,16.3,...
08:20,0,4.9,11132.15,-273.17,16.35,...
```

Python code

```
import pandas as pd
from datetime import datetime

df = pd.read_csv('data_pv_15-01-2013.csv')
irr = df['IntSolIrr']

# 4.1 maximum
max_irr = irr.max()
max_time = df.loc[irr.idxmax(), 'TimeStamp']

# 4.2 average, full 24 h
avg_irr_24h = irr.mean()

# 4.3-4.6: use a small threshold, not just > 0 (see note below)
THRESHOLD = 1.0
daylight = df[irr > THRESHOLD]
t_start = daylight['TimeStamp'].iloc[0]
t_end   = daylight['TimeStamp'].iloc[-1]

t0 = datetime.strptime(t_start, '%H:%M')
t1 = datetime.strptime(t_end, '%H:%M')
duration_h   = (t1 - t0).total_seconds() / 3600
duration_pct = duration_h / 24 * 100
avg_irr_daylight = daylight['IntSolIrr'].mean()
```

Why the threshold matters — actual data from this day

Filtering with the naive condition IntSolIrr > 0 gives a first non-zero reading at 00:05 (value 0.03 W/m^2) — clearly not sunrise. Several more near-zero, non-zero readings appear scattered through the night:

```
TimeStamp   IntSolIrr
00:05       0.03
00:35       0.29
00:40       0.16
00:45       0.06
00:50       0.10
01:00       0.31
02:35       0.03
04:20       0.03
```

These are sensor-noise artifacts, not sunlight. Raising the threshold to `IntSolIrr > 1.0` removes all of them while barely affecting the real sunrise/sunset times, because the true morning ramp jumps well past 1 W/m² within a single 5-minute sample:

Threshold	First time	Last time	Samples counted
> 0 W/m² (naive)	00:05	17:30	122
> 1 W/m² (used)	08:10	17:25	112

Results

Quantity	Value
4.1 Maximum solar irradiance	520.73 W/m² (at 12:00)
4.2 Average irradiance, full 24 h	48.39 W/m²
4.3 Earliest time with radiation	08:10
4.4 Latest time with radiation	17:25
4.5 Duration of sunlight	9.25 h (38.54% of 24 h)
4.6 Average irradiance during daylight	124.41 W/m²
Bonus: total daily insolation	1161.3 Wh/m² (1.161 kWh/m²)

Chart

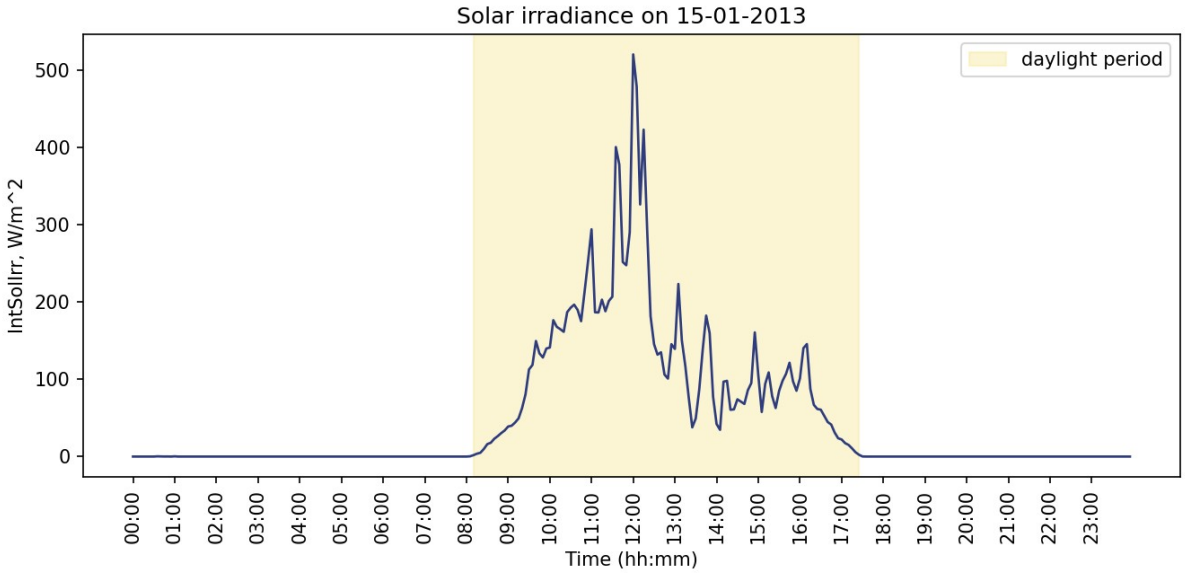

Figure 1. Solar irradiance (IntSolIrr) on 15-01-2013, with the detected daylight period shaded. The repeated rises and dips around midday are consistent with passing clouds.

Interpretation

The curve peaks sharply at solar noon (12:00) but is far from a smooth dome — the many secondary peaks and dips between about 10:00 and 16:30 point to intermittent cloud cover rather than a clear sky. The gap between the daily maximum (520.73 W/m²) and the daylight-average (124.41 W/m²) is large for exactly this reason: a clear-sky January day in this location would show a smoother curve and a higher average relative to its peak. This is a good example of why a single summary statistic (the maximum, or even the mean) is not enough to describe a day — the shape of the curve itself carries information the numbers alone do not.

Control questions

- Why is it important to check a dataset's documentation (an Info sheet, a codebook) before using a column, rather than assuming its name tells you everything? What went wrong in this dataset if you had used column B instead of column C?
- What is the practical difference between filtering with IntSolIrr > 0 and IntSolIrr > 1 in this dataset? Which one gives a physically meaningful sunrise time, and why?
- Why does pandas' vectorized .max() / .mean() scale better than a hand-written loop as the dataset grows larger?
- How would you compute the total energy received per square meter over a day, starting from 5-minute irradiance readings in W/m²?
- What could explain the multiple peaks and dips in irradiance seen around midday in the worked example?
- If your assigned day had zero valid daylight readings after filtering, what would that suggest about the data, and how would you check?