LECTURE 4

Relationships Between Variables and Correlation

Scatter plots • violin & box plots • covariance & Pearson's r • Spearman's rank correlation • correlation vs. causation • simple linear regression

Principles and Methods of Data Analysis in the Energy Industry

Think Stats, Ch. 7 • Elements of Data Science, Ch. 9

WHERE WE ARE

From single variables to relationships between them

A

Scatter Plots & Overplotting

Seeing a relationship, and fixing the plot when points pile up

B

Violin & Box Plots

Comparing a distribution across many groups at once

C

Quantifying Correlation

Covariance, Pearson's r, Spearman's rank correlation, and their limits

D

Simple Linear Regression

Fitting a line, and why correlation strength is not practical importance

Running example: the GECAD photovoltaic dataset from Lectures 1–3 and Labs 1–2 — solar irradiance, module temperature, AC power and grid frequency, now examined together instead of one at a time.

LEARNING OBJECTIVES

By the end of this lecture, you will be able to

- Build a scatter plot of two energy variables and recognize when overplotting is hiding the true shape of the relationship.
- Fix overplotting using transparency (alpha), smaller markers, and jittering.
- Compare a distribution across many categories at once using violin plots and box plots.
- Compute and interpret covariance, standardized (z-)scores, and Pearson's correlation coefficient.
- Explain why Pearson's r can be misleading for nonlinear relationships or in the presence of outliers, and when Spearman's rank correlation is the safer choice.
- Distinguish correlation from causation, and recognize a spurious correlation created by a shared on/off pattern in two signals.
- Fit and interpret a simple linear regression with scipy.stats.linregress, and explain why a strong correlation does not automatically mean a practically important slope.

PART A

Scatter Plots & Overplotting

Seeing a relationship between two energy variables

One variable at a time is not the whole story

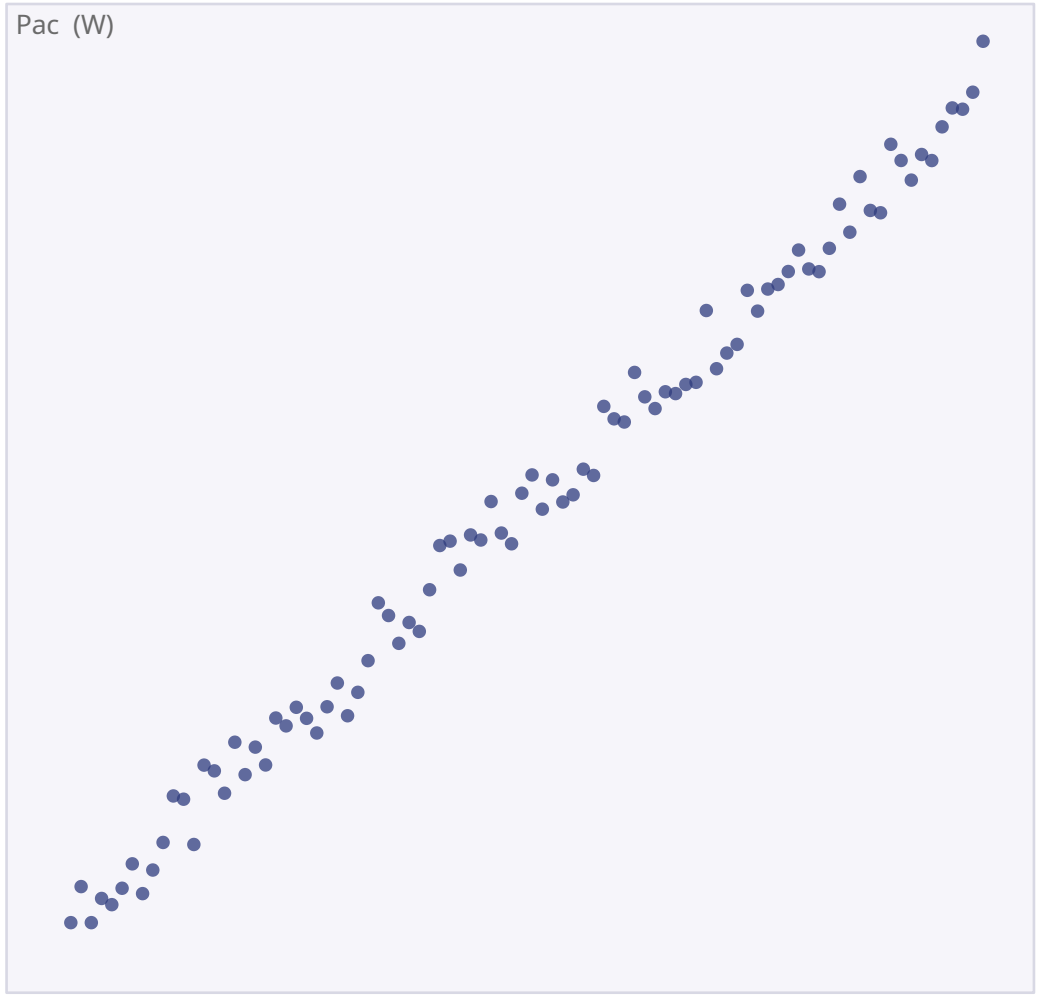
- Lectures 1–3 treated each column on its own: a histogram of solar irradiance, a PMF of module temperature, a CDF of AC power.
- But an engineer rarely cares about a single sensor in isolation — the real question is how two signals move together:
  - Does AC power track solar irradiance the way the datasheet predicts?
  - Does module temperature rise with irradiance — and by how much?
  - Does the inverter's output frequency depend on how much power it is producing, or is that just an illusion?
- A scatter plot is the simplest tool for this: one point per observation, one variable on each axis.

SCATTER PLOTS

Plotting AC power against solar irradiance

```
import matplotlib.pyplot as plt

plt.plot(df['IntSolIrr'], df['Pac'], 'o',
         markersize=3, alpha=0.5)
plt.xlabel('Solar irradiance, W/m^2')
plt.ylabel('AC power, W')
plt.title('Pac vs. IntSolIrr -- 15-01-2013')
```

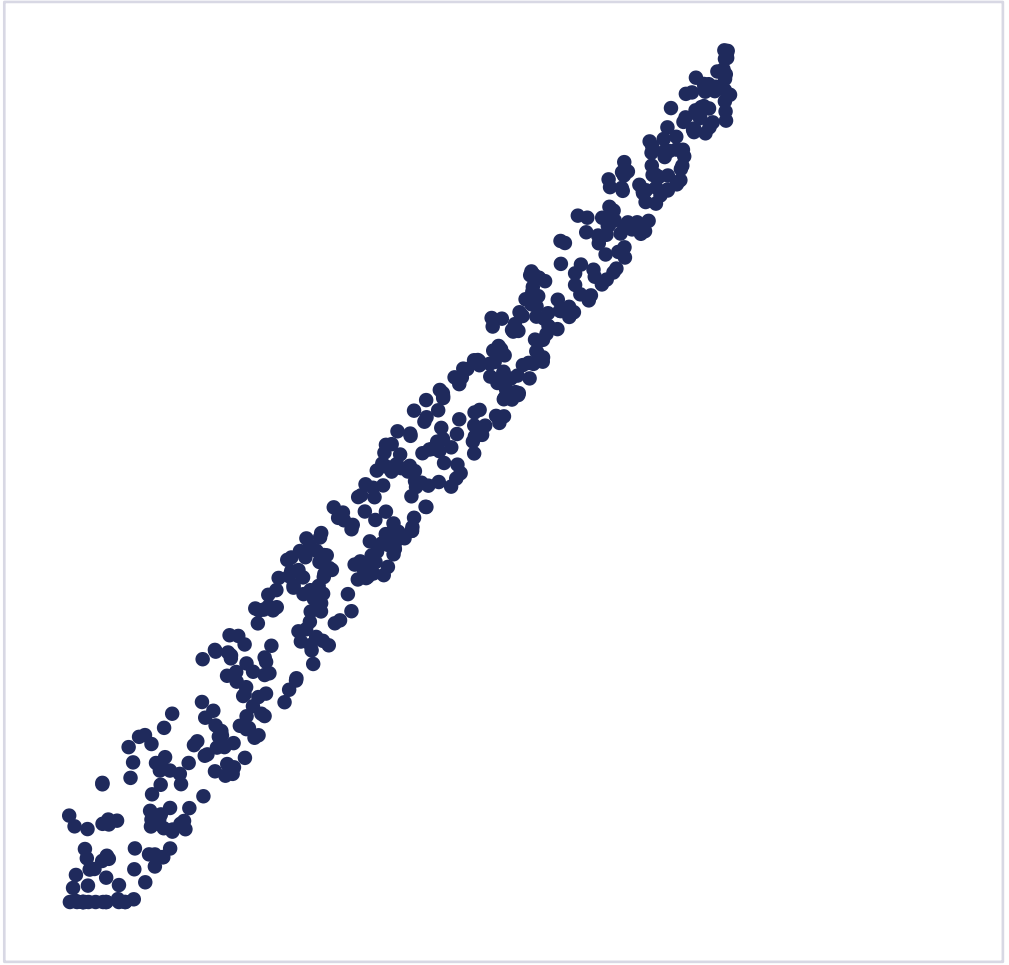


- Each of the 288 five-minute samples from one day becomes one point.
- plt.plot(..., 'o') draws markers with no connecting line — the right choice whenever the x-order (here, time) is not the axis you are plotting.
- Even from one day's data the pattern is unmistakable: power rises almost in lock-step with irradiance.

THE OVERPLOTTING PROBLEM

One day is fine. Twenty-five days is a different story.

- Combine all 25 available January sheets and the same plot has 6,350 points instead of 288.
- With solid, full-size markers the dense region — low irradiance, low power, night-time and cloudy readings — turns into a single black blob.
- You lose exactly the information a scatter plot exists to show: how densely the points are packed, and whether there is structure inside that dense region.
- This is overplotting, and it gets worse with every additional day, sensor, or turbine you add to the dataset.



6,350 points, opaque markers

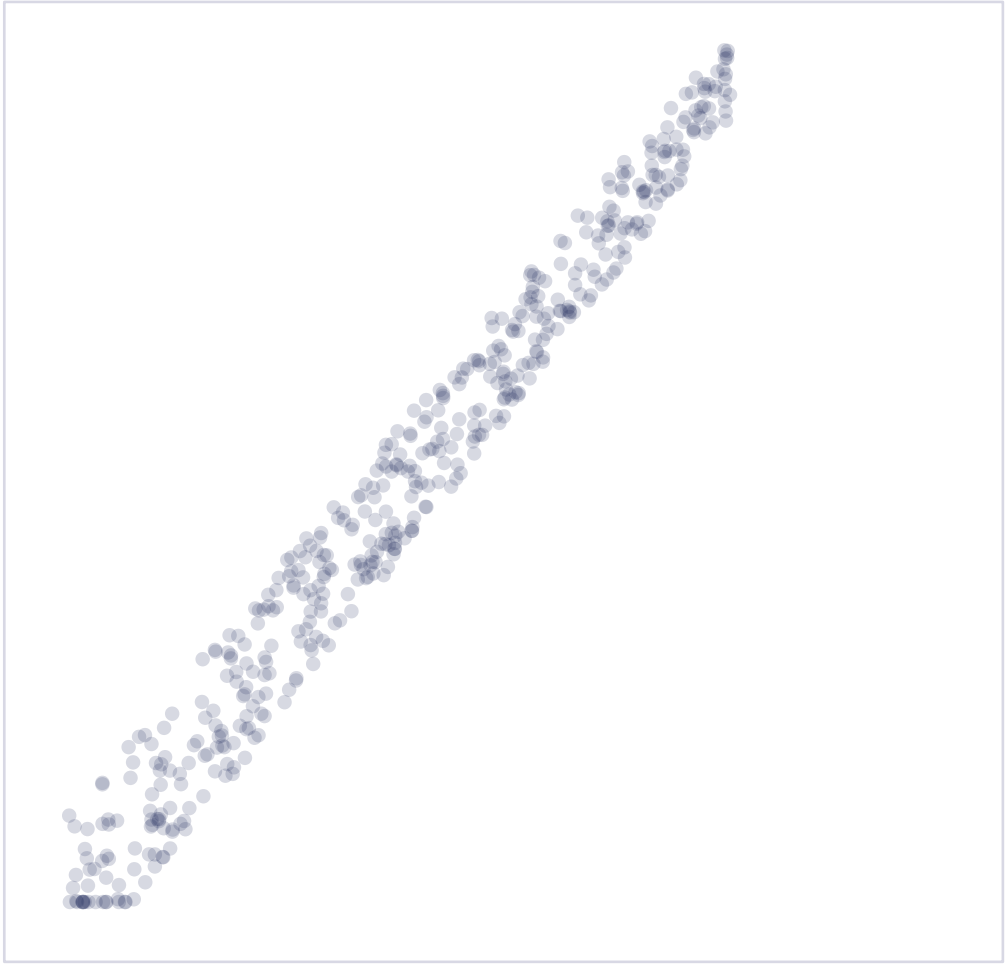
Elements of Data Science, Ch. 9 walks through this exact progression using survey data on height and weight — we use the same three fixes here on sensor data instead.

FIX 1 OF 3 — TRANSPARENCY

alpha turns overlap into visible density

```
plt.plot(all_days['IntSolIrr'], all_days['Pac'],
         'o', alpha=0.03)
```

- alpha sets each marker's opacity from 0 (invisible) to 1 (solid).
- With thousands of overlapping points, a small alpha (here 0.03) means a spot only looks dark where MANY points really do overlap.
- The dense night-time / low-power region reappears as a visibly darker patch, rather than a single flat mass — you can see that it is dense, and roughly how dense.
- Rule of thumb: start around alpha = 1/√n of your point count and adjust by eye.



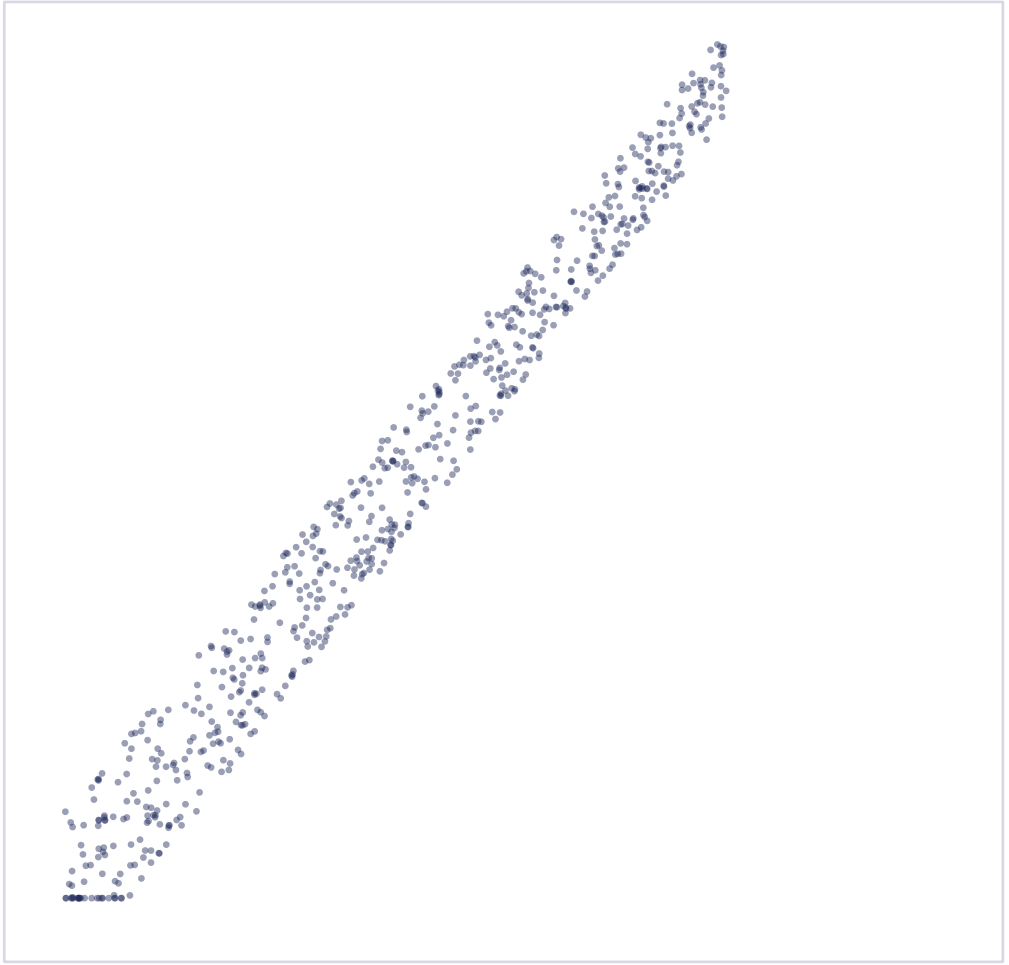
alpha = 0.03

FIX 2 OF 3 — SMALLER MARKERS

markersize reduces overlap in the first place

```
plt.plot(all_days['IntSolIrr'], all_days['Pac'],
         'o', markersize=0.8, alpha=0.1)
```

- Smaller markers overlap less to begin with, so you need less transparency to see structure — the two fixes are normally combined.
- Watch for the failure mode in the other direction: markers so small (and alpha so low) that sparse, IMPORTANT points — like a rare fault reading — disappear completely.
- There is no single correct setting; you are trading off visibility of the dense region against visibility of rare points, and that trade-off should match what the plot is for.



markersize = 0.8, alpha = 0.1

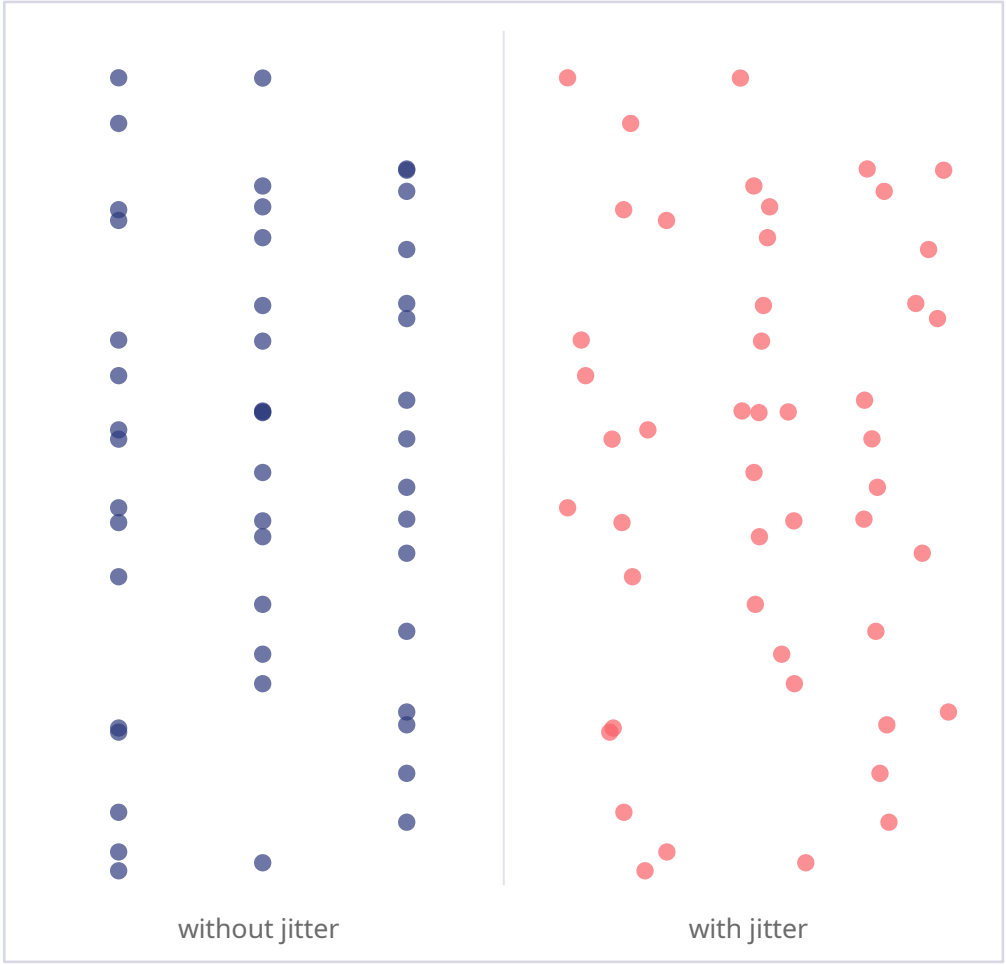
FIX 3 OF 3 — JITTERING

Adding noise to break up artificial stacking

```
import numpy as np

jittered_hour = (hour_of_day +
                 np.random.normal(0, 0.15, n))
plt.plot(jittered_hour, df['Pac'], 'o',
         markersize=2, alpha=0.1)
```

- Some variables are recorded coarsely — the hour of day, or a sensor that rounds to whole degrees — so many real observations land on EXACTLY the same x (or y) value and stack into a vertical (or horizontal) line.
- Jittering adds a small amount of random noise (np.random.normal(0, small_sd, n)) purely for display, spreading the stack sideways so the eye can judge density again.
- Jittering never changes the underlying data or any calculation — only the copy used for the plot.



PART B

Violin & Box Plots

Comparing a distribution across many groups at once

Why not just draw 24 histograms?

- Suppose you want to see how AC power output is distributed for EACH hour of the day — 24 separate groups.
- Overlaying 24 histograms is unreadable, and 24 separate small histograms are hard to compare side by side.
- What you usually want is a compact summary of each group's distribution, lined up so groups can be compared directly — that is exactly what violin and box plots give you.
- Both take a categorical grouping variable (here, hour of day: 0–23) on one axis and a numeric variable (Pac) on the other.

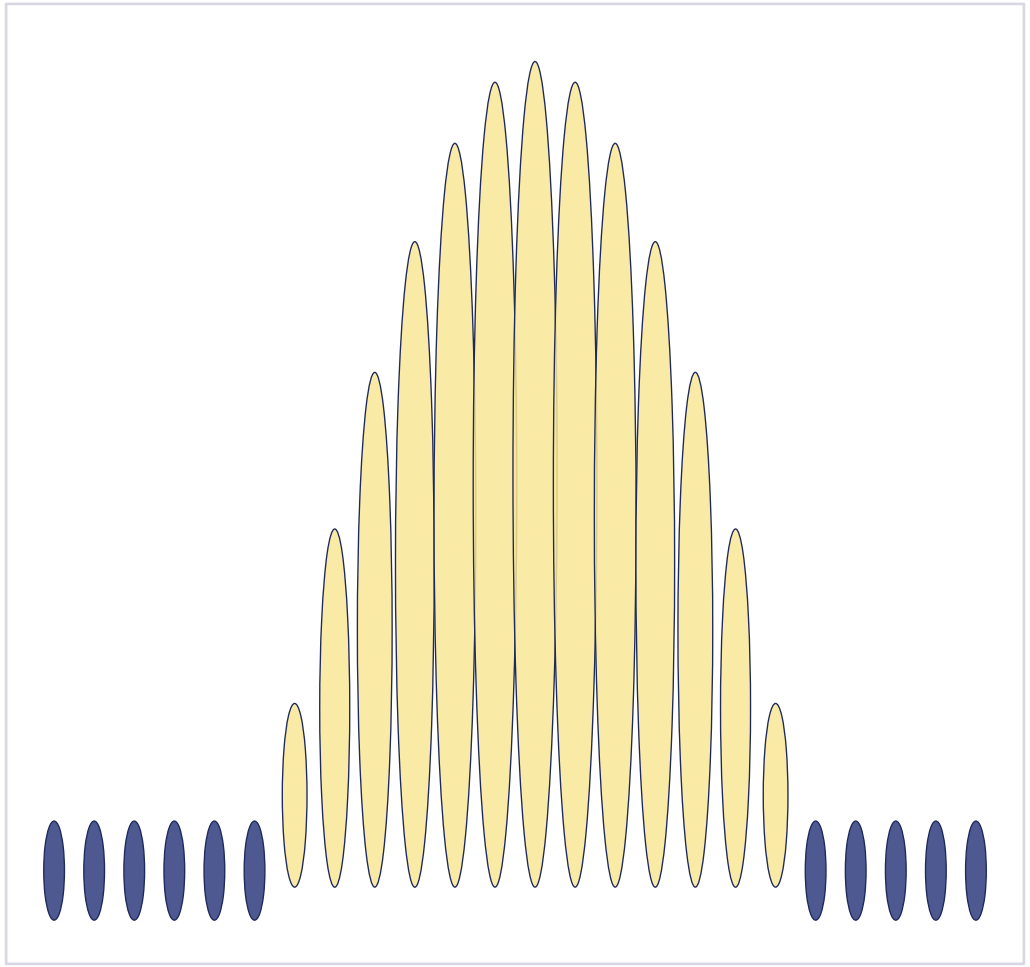
VIOLIN PLOTS

The full shape of each group's distribution

```
import seaborn as sns

sns.violinplot(x='hour', y='Pac',
               data=all_days, inner=None)
```

- A violin plot draws a mirrored, smoothed density estimate (like the KDE from Lecture 3) for each group, so its WIDTH at any height shows how common that value is.
- Reading the hourly Pac violins: the plot for hour 3 is a thin sliver pinned at 0 W (the panel is idle at night); the plot for hour 12 is wide and spread from 0 up past 150 W (full range of possible cloud conditions at midday).
- inner=None removes the default inner box/quartile marks when you only want the shape; leave it in when you also want the quartiles at a glance.

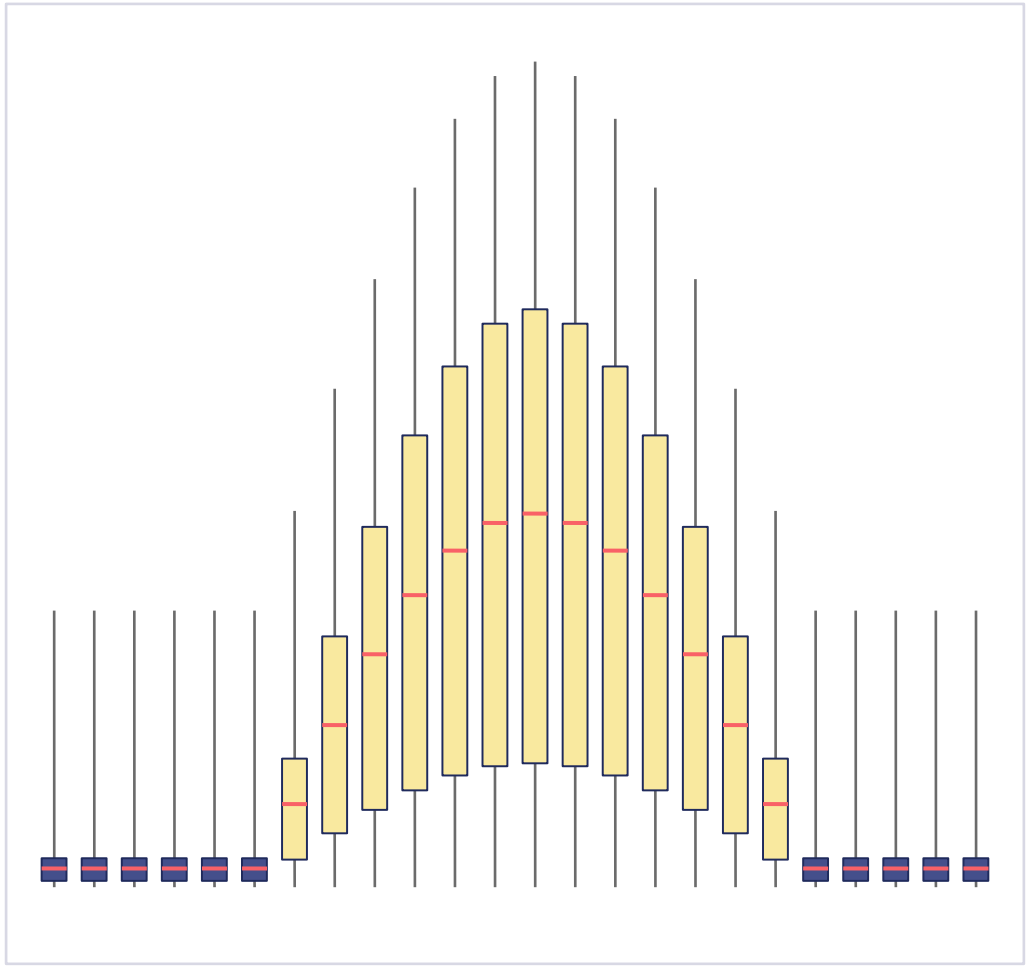


Pac by hour of day (24 groups)

A compact five-number summary per group

```
sns.boxplot(x='hour', y='Pac',
            data=all_days, whis=10)
```

- Each box shows the median (line), the interquartile range / IQR (box), and whiskers extending to the most extreme non-outlier points; points beyond the whiskers are drawn individually as outliers.
- whis controls how far the whiskers reach before a point counts as an outlier (default 1.5 × IQR). A busy dataset with many legitimate extreme values — like a solar plant on a rare, unusually clear day — often needs a larger whis (e.g. 10) so genuine variation is not flagged as noise.
- A box plot is more compact than a violin but hides multi-peaked (bimodal) shapes — use both together when in doubt.
- For a right-skewed variable such as Pac, plotting the y-axis on a log scale (plt.yscale('log')) often makes the boxes easier to compare.



Pac by hour of day — boxes

WORKED EXAMPLE

Pac by hour of day, across all 25 January sheets (n = 6,350)

- Data: every 5-minute reading from all 25 available day-sheets, combined and tagged with its hour of day (0–23).
- Hours 0–5 and 20–23 (night): Pac is essentially a spike at 0 W — the violin is a thin sliver, the box collapses to a line.
- Hours 10–14 (midday): Pac spans the widest range — from 0 W on a heavily overcast day up to the daily maximum on a clear one. The box's IQR sits high, but the whisker (with whis=10) still reaches down toward 0.
- The transition hours (7–9 and 15–17, sunrise/sunset) show the fastest-growing and fastest-shrinking spread — exactly what you would expect physically.

Takeaway: violin and box plots let you see a whole day's generation profile — not just its average — in a single figure, which is exactly what a 24-panel grid of histograms could not give you at a glance.

PART C

Quantifying Correlation

From “they look related” to a single number

COVARIANCE

Do two variables move together, or in opposite directions?

- Covariance measures whether two variables tend to be above their own means at the same time, or not:

$Cov(X, Y) = mean[(x_i - mean(x)) \times (y_i - mean(y))]$

- Positive covariance: when irradiance is above its average, power tends to be above its average too.
- Negative covariance: when one variable is above its average, the other tends to be BELOW its average (e.g., module efficiency tends to dip when module temperature is unusually high).
- Problem: covariance's units are the PRODUCT of the two variables' units (here, W/m^2 × W), so its raw size tells you nothing about strength — you cannot compare a covariance of 5,000 to a covariance of 12 without knowing the units involved.

Removing units by rescaling each variable

```
z_irr = (df['IntSolIrr'] - df['IntSolIrr'].mean()) / df['IntSolIrr'].std()
z_pac = (df['Pac'] - df['Pac'].mean()) / df['Pac'].std()
```

- A z-score (standard score) expresses how many standard deviations a value is above or below its own mean: z = (x − mean) / std.
- After standardizing, EVERY variable has mean 0 and standard deviation 1 — W/m^2, watts, hertz, degrees all become directly comparable, unit-free numbers.
- Standardizing is the trick that turns covariance into a number you can actually interpret: computing the covariance of the standardized variables gives Pearson's correlation coefficient.

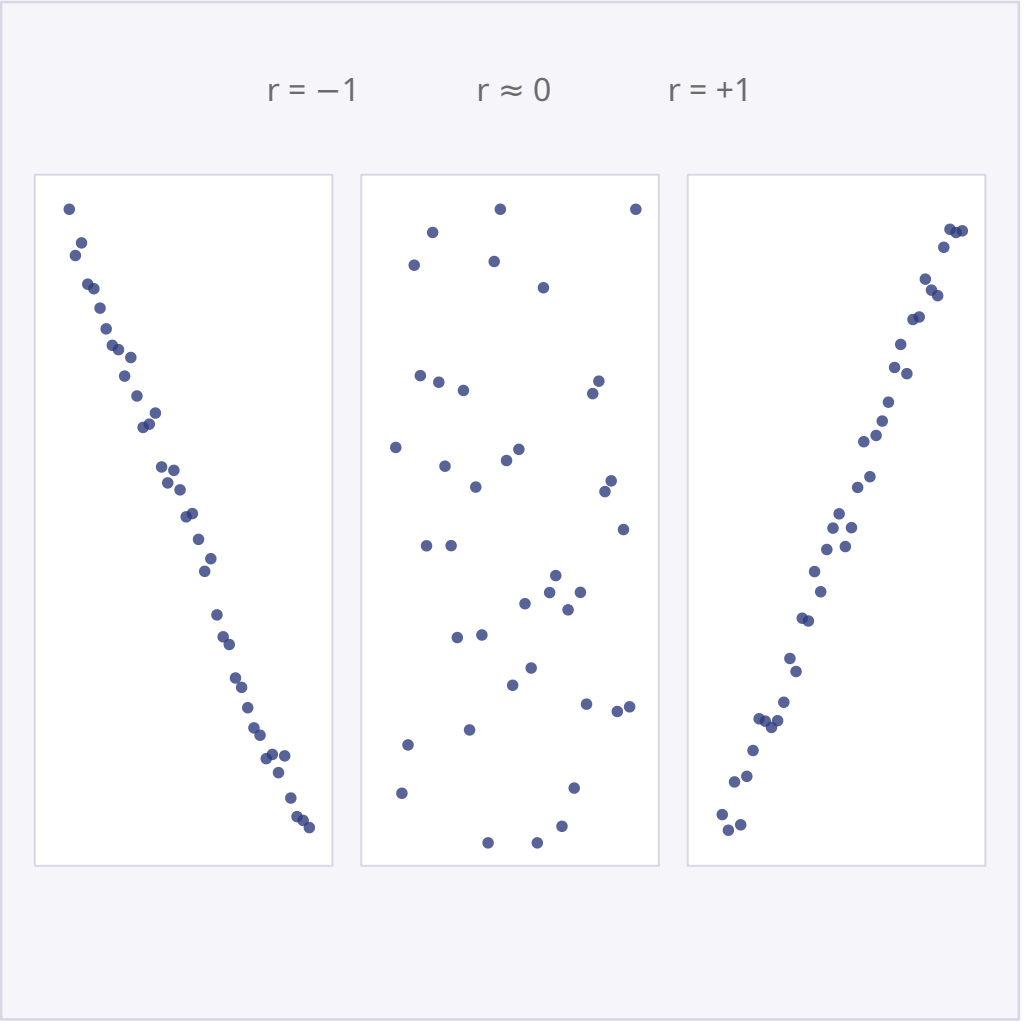
PEARSON'S CORRELATION COEFFICIENT

One number, always between −1 and +1

```
df[['IntSolIrr', 'TmpMdul C', 'Pac', 'Fac']].corr()

import numpy as np
np.corrcoef(df['IntSolIrr'], df['Pac'])[0, 1]
```

- r = Cov(X, Y) / (std(X) × std(Y)) — the covariance of the standardized variables.
- r = +1: a perfect increasing straight-line relationship. r = −1: perfect decreasing straight line. r = 0: no LINEAR relationship (not necessarily “no relationship at all” — see the limitation in a moment).
- |r| above about 0.7 is usually called “strong”, 0.3–0.7 “moderate”, and below 0.3 “weak” — but always look at the scatter plot too; a single number can hide a lot.



WORKED EXAMPLE

Correlation matrix, GECAD PV panel — 15-01-2013 (n = 288)

	IntSolIrr	TmpMdul C	Pac	Fac
IntSolIrr	1.000	0.883	0.997	0.721
TmpMdul C	0.883	1.000	0.879	0.718
Pac	0.997	0.879	1.000	0.678
Fac	0.721	0.718	0.678	1.000

- Pac and IntSolIrr are almost perfectly linearly related (r = 0.997) — exactly what a solar inverter's operating principle predicts.
- TmpMdul C rises with IntSolIrr too (r = 0.883), but less tightly — module temperature depends on more than just sunlight (ambient conditions, thermal mass, cooling).
- Fac (grid frequency) correlates with everything at r ≈ 0.68–0.72 — suspiciously similar values across very different physical quantities. The next few slides show why that number is misleading.

LIMITATION 1 — NONLINEAR RELATIONSHIPS

A wind turbine's power curve breaks Pearson's r

```
# Simulated turbine: P is proportional to wind speed cubed
wind_speed = np.linspace(3, 14, 200)   # m/s, cut-in to rated
power = 0.5 * wind_speed**3 + noise
np.corrcoef(wind_speed, power)[0, 1]   # -> 0.93 (looks strong)

# But look closer: below ~5 m/s the turbine barely turns,
# and above ~12 m/s the curve flattens (rated power) --
# neither region is captured by a single straight-line slope.
```

- A power curve P ∝ v^3 is strongly, deterministically related to wind speed — but it is a CURVE, not a line.
- Pearson's r can still come out high on a smooth curve like this one (because it is monotonically increasing), but it systematically understates the relationship near the cut-in and rated-power plateaus, where power barely changes even though wind speed does.
- The textbook worst case: a symmetric parabola y = x^2 + noise gives Pearson's r ≈ 0 even though y is completely determined by x — correlation measures LINEAR association only, and can miss a relationship entirely.

LIMITATION 2 — OUTLIERS

A handful of extreme points can dominate r

- Because Pearson's r is built from means and standard deviations, it is sensitive to exactly the same kind of extreme values that distort a mean (Lecture 2).
- A single misrecorded sensor spike — a momentary voltage transient logged as a huge Pac value, for instance — can swing a correlation coefficient noticeably, even though it says nothing about the true relationship between the variables.
- Practical defense: always look at the scatter plot alongside the number, and consider computing r with and without suspected outliers to see how much it moves.
- This is also where a threshold-based cleaning step — like the sensor-noise threshold used for daylight detection in Lab 1 — protects a correlation analysis, not just a mean.

SPEARMAN'S RANK CORRELATION

Correlate the RANKS, not the raw values

```
from scipy.stats import spearmanr

rho, p_value = spearmanr(
    df['IntSolIrr'], df['Pac'])
```

- Spearman's rho replaces each value with its RANK within its own variable, then computes Pearson's r on the ranks.
- Because it only cares about ORDER, Spearman's rho is robust to outliers and correctly detects any MONOTONIC relationship — not only a linear one.
- It is the more honest choice whenever you suspect the true relationship curves (like a turbine power curve) or the data contains occasional bad readings.
- Rule of thumb: compute both. If Pearson's r and Spearman's rho are close, the relationship is close to linear; if they differ substantially, that gap is itself informative.

On the single day 15-01-2013: IntSolIrr vs. Fac gives Pearson r = 0.721 but Spearman ρ = 0.90 — a large gap that is a warning sign, explored on the next slide.

A real, cautionary example from this course's own dataset

- Fac (grid frequency) correlated with IntSolIrr at r = 0.72 on slide 20 — strong enough to tempt a claim that sunlight affects grid frequency.
- Look closer at WHY: the inverter reports Fac = 0 exactly whenever it is switched off (no sunlight → no generation → idle), and a near-constant ~50.00 Hz whenever it is switched on.

Restrict the comparison to the moments the inverter is actually online (Fac > 0) — all 288 × 25 = 6,350 readings, 2,137 of them with Fac > 0:

Pearson r(IntSolIrr, Fac | Fac > 0) = −0.02. Essentially zero.

- The original 0.72 was never about physics — it was a SHARED ON/OFF PATTERN: both variables happen to be “low” at night and “high” in daytime for unrelated reasons (irradiance because of the sun; Fac because the inverter is running at all). This is a classic confound.

PART D

Simple Linear Regression

Fitting a line — and reading its slope honestly

SCIPY.STATS.LINREGRESS

Fitting the best straight line through a scatter

```
from scipy.stats import linregress

result = linregress(df['IntSolIrr'], df['Pac'])
result.slope, result.intercept, result.rvalue,
result.pvalue, result.stderr
```

- linregress fits y = slope × x + intercept by least squares (minimizing the sum of squared vertical distances from each point to the line).
- rvalue is exactly Pearson's r for the two variables — regression and correlation are two views of the same linear relationship.
- rvalue**2 (“R-squared”) is the fraction of the variance in y that is explained by a linear function of x.
- pvalue tests the null hypothesis that the true slope is zero — useful, but with thousands of sensor readings, even a tiny, practically unimportant slope will often be “statistically significant.”

WORKED EXAMPLE

Pac ~ IntSolIrr, across all 25 days (n = 6,350)

Subset	slope (W per W/m^2)	intercept (W)	r	r^2
All readings	0.248	−0.37	0.870	0.757
Daylight only (IntSolIrr > 1)	0.250	−1.33	0.828	0.686
Module temp. ~ irradiance (daylight)	0.021 °C per W/m^2	16.78	0.705	0.497

- About 0.25 W of AC output per additional W/m^2 of irradiance, whether or not the (mostly-zero) night readings are included — the relationship is dominated by the daytime data either way.
- r^2 ≈ 0.69–0.76: roughly three-quarters of the variation in output power is explained by irradiance alone across a whole month of mixed weather; the rest comes from cloud transients, temperature derating, and inverter behavior.
- Module temperature climbs more gently with irradiance (r^2 ≈ 0.50) — a real, moderate, but far less tight relationship than power itself.

CORRELATION STRENGTH IS NOT PRACTICAL IMPORTANCE

The same r can hide very different real-world slopes

Dataset	correlation r	slope	effect over 30 years
Site A — panel degradation vs. age	0.758	0.019 %/yr	0.56 % lost over 30 yr
Site B — panel degradation vs. age	0.478	0.176 %/yr	5.29 % lost over 30 yr

- This pattern (adapted from Think Stats' two “fake dataset” example, reframed here for panel-degradation monitoring) is a real trap: Site A has the STRONGER correlation, but Site B has the more important, more expensive practical effect.
- The reason: correlation reflects how TIGHTLY points cluster around a line, while the regression slope reflects how STEEP that line is — two independent properties of a scatter plot.
- A noisier relationship (lower r, more scatter around the line) can still carry a much larger, more consequential slope than a very tight, clean-looking one — always report both numbers, never r alone.

WRAP-UP

From “related” to “how, how strongly, and why”

- Scatter plots reveal a relationship; alpha, markersize, and jittering keep them readable at real-world data volumes.
- Violin and box plots compress many groups' distributions into one comparable figure.
- Covariance, z-scores, and Pearson's r turn “looks related” into a single, comparable number — but one that only captures LINEAR association and is sensitive to outliers.
- Spearman's rank correlation, a look at the scatter plot, and healthy suspicion of shared on/off patterns all guard against being misled.
- Simple linear regression fits and quantifies that line — and correlation strength is not the same question as “does the slope matter.”

Coming up in Lecture 5: from describing relationships to estimation and hypothesis testing — how confident can we be in a correlation or a slope computed from a sample of readings?