BP201T · B.PHARM THEORY

BP201T Applied Biostatistics and Data Analytics for Pharmaceutical Sciences

Unit V — Statistical Analysis Using Python

Final Unit Notes

Pills & Python Notes

From-zero teaching standard: every technical term is defined before use.

Syllabus coverage: Unit V (6 hours).

PILLS & PYTHON

Pharmacy • Data • AI • Healthcare

Instagram @pillsandpython · pillsandpython.com · YouTube @PillsAndPython

Chapter 5 — Statistical Analysis Using Python: Case-Based Learning (Unit V, 6 hours)

5.1 Learning objectives

After studying this chapter, you will be able to:

1. Run a complete descriptive analysis on a pharmaceutical dataset with Pandas.
2. Compute correlation with SciPy and interpret the coefficient and its p-value.
3. Fit a simple linear regression with Statsmodels and with Scikit-learn.
4. Read an OLS output summary: coefficients, standard errors, p-values, R-squared.
5. Write a short, honest statistical report of your findings.

5.2 The case dataset (illustrative)

A small illustrative trial dataset — 12 patients, antihypertensive drug:

Patient	dose_mg	age_yrs	bp_drop (mmHg)
1	10	45	4
2	15	52	7
3	20	38	8
4	25	61	11
5	30	49	12
6	35	55	14
7	40	42	15
8	45	58	18
9	50	47	19
10	55	63	21
11	60	51	22
12	65	44	24

This dataset is illustrative — invented only to teach the workflow, not real trial data.

5.3 Case step 1 — descriptive statistics with Pandas

```
import pandas as pd

df = pd.DataFrame({
    "dose_mg": [10,15,20,25,30,35,40,45,50,55,60,65],
    "age_yrs": [45,52,38,61,49,55,42,58,47,63,51,44],
    "bp_drop": [4,7,8,11,12,14,15,18,19,21,22,24],
})
print(df.describe().round(2))
```

Line-by-line: build the DataFrame from three lists; .describe() computes count, mean, std, min, quartiles, max for every numeric column; .round(2) tidies decimals.

Sample output (computed by running the code):

```
       dose_mg  age_yrs  bp_drop
count    12.00    12.00    12.00
mean     37.50    50.42    14.58
std      18.03     7.75     6.39
min      10.00    38.00     4.00
25%      23.75    44.75    10.25
50%      37.50    50.00    14.50
75%      51.25    55.75    19.50
max      65.00    63.00    24.00
```

Interpretation for the report: 12 patients, mean age ~50 years, mean BP drop ~14.6 mmHg (SD 6.4). Doses span 10–65 mg evenly (mean = median = 37.5 — symmetric by design). No missing values (count = 12 everywhere).

Key takeaway: `.describe()` is always step one — it checks data quality (missing values, weird ranges) and gives the story's headline numbers.

5.4 Case step 2 — correlation analysis with SciPy

```
from scipy import stats

r, p = stats.pearsonr(df["dose_mg"], df["bp_drop"])
print("r =", round(r, 3), " p =", p)
```

Line-by-line: stats.pearsonr(x, y) returns two things — the Pearson r and the p-value testing H_0: "true correlation is zero". (Age vs BP drop gives r ≈ 0.22 — weak, as the matrix below shows.)

Sample output (values computed from the data):

```
r = 0.997   p < 0.0001
```

Interpretation: r = 0.997 is an extremely strong positive correlation; p < 0.0001 means such a strong link would almost never appear by chance if dose and response were truly unrelated — reject H_0. The full correlation matrix (from df.corr()) is:

```
          dose_mg  age_yrs  bp_drop
dose_mg     1.000    0.161    0.997
age_yrs     0.161    1.000    0.224
bp_drop     0.997    0.224    1.000
```

Age barely correlates with anything (r ≈ 0.16–0.22) — dose is the driver.

Key takeaway: SciPy hands you r *and* its p-value together — strength and evidence in one line.

5.5 Case step 3 — linear regression with Statsmodels

Statsmodels is the library for *statistical* modelling — its summaries read like a textbook page.

```
import statsmodels.formula.api as smf

model = smf.ols("bp_drop ~ dose_mg", data=df).fit()
print(model.summary())
```

Line-by-line: `smf.ols("bp_drop ~ dose_mg", data=df)` declares the model "predict bp_drop from dose_mg" (`~` reads as "modelled by"); `.fit()` finds the least-squares line; `.summary()` prints the full statistical report.

How to read the summary (the four things that matter):

1. **Coefficients table** — the fitted line from our data: intercept ≈ **1.34**, slope ≈ **0.353**, i.e. **bp_drop = 1.34 + 0.353 × dose**. Each extra mg predicts ~0.35 mmHg more BP drop.
2. **P>|t| column** — the p-value for each coefficient. Here both are far below 0.05: the slope is statistically significant; dose genuinely predicts response.
3. **R-squared ≈ 0.993** — the model explains 99.3% of the variation in BP drop. ($R^2 = r^2 = 0.997^2$.) Near 1 = the line fits beautifully.
4. **nobs = 12** — always check the sample size; great R^2 on 12 illustrative points is a teaching demo, not a drug approval.

Key takeaway: The summary answers three questions: *what* is the equation (coef), *is it real* (p-values), and *how well does it fit* (R-squared).

5.6 Case step 3 (alternative) — regression with Scikit-learn

Scikit-learn is the library for *machine learning* — less statistical detail, built for prediction pipelines.

```
from sklearn.linear_model import LinearRegression

X = df[["dose_mg"]]          # predictors must be a 2-D table
y = df["bp_drop"]            # outcome is a single column
reg = LinearRegression().fit(X, y)   # fit the line
print("Slope:", reg.coef_, " Intercept:", reg.intercept_)
print("R²:", reg.score(X, y))
print("Predicted drop at 70 mg:", reg.predict([[70]]))
```

Line-by-line: wrap the dose column in double brackets (sklearn needs 2-D input); `.fit(X, y)` learns the line; `.coef_` / `.intercept_` reveal it; `.score` returns R^2; `.predict([[70]])` forecasts a new dose.

Sample output (same least-squares maths as Statsmodels):

```
Slope: [0.3531]  Intercept: 1.3403
R²: 0.9932
Predicted drop at 70 mg: [26.06]
```

Interpretation: Identical line (0.353 slope, 1.34 intercept) — the libraries agree, as they should. But note: 70 mg lies *outside* our 10–65 mg data range, so 26.06 mmHg is an **extrapolation** — flag it as uncertain, never as a promise.

Statsmodels vs Scikit-learn — which to use?

	Statsmodels	Scikit-learn
Strength	Full statistical summary (p-values, CIs, R^2)	Prediction pipelines, machine learning
Use when	You must *explain and justify* the model	You must *predict* new cases at scale
Exam tip	"Interpret the output" → Statsmodels	"Predict / deploy" → Scikit-learn

Key takeaway: Same line, different jobs — Statsmodels explains, Scikit-learn predicts.

5.7 Case step 4 — writing the statistical report

A good report is short and honest. Structure:

1. **Objective** — one line: what question the analysis answers.
2. **Data & methods** — dataset description (n, variables), methods used (descriptive stats, Pearson r, OLS regression), software (Python, Pandas/SciPy/Statsmodels).
3. **Results** — numbers with interpretation: "Mean BP drop was 14.6 mmHg (SD 6.4). Dose and BP drop correlated strongly ($r = 0.997$, $p < 0.0001$). The fitted line was bp_drop = 1.34 + 0.353 × dose ($R^2 = 0.993$); the slope was statistically significant ($p < 0.05$)."
4. **Limitations** — small illustrative sample (n = 12); association is not causation; prediction at 70 mg is extrapolation.
5. **Conclusion** — one careful line: "Higher doses were associated with greater BP reduction in this illustrative dataset; confirmatory trials are needed."

Golden rules: report effect sizes with p-values (never p-values alone); state limitations plainly; never claim causation from correlation; mark illustrative data as illustrative.

Key takeaway: A report = question → methods → numbers → honest limits → careful conclusion.

5.8 Pharmacy relevance (Unit V)

- **Trial analysis:** this exact workflow (describe → correlate → regress → report) is how Phase II dose-finding results are analysed.
- **QC trending:** regressing assay values over months detects drift before batches fail.
- **Regulatory writing:** agencies expect p-values, CIs, and clear limitations — exactly this report structure.

5.9 Common confusions

1. **Statsmodels vs Scikit-learn** output: same line, different detail — don't panic when numbers match but layouts differ.
2. **R^2 vs r:** R^2 (0–1) is the *fraction of variation explained*; r (−1 to +1) is the correlation. In simple regression, $R^2 = r^2$.
3. **p < 0.05 on the slope** means the relationship is statistically real — not that it is clinically big.
4. **Prediction vs explanation:** a model can predict well (high R^2) without proving *why*.

5.10 Common mistakes

- Forgetting `df[["dose_mg"]]` needs double brackets in sklearn (single brackets crash `.fit`).
- Reporting the regression equation without R^2 or p-values.
- Extrapolating beyond the data range without a warning.
- Presenting illustrative/demo data as if it were real trial evidence.

5.11 One-minute revision (Unit V)

- Workflow: **describe** → **correlate** → **regress** → **report**.
- Pandas `.describe()` = data health-check + headline numbers.
- `scipy.stats.pearsonr` → r **and** its p-value.
- Statsmodels `ols("y ~ x").fit().summary()` → coefficients, **p-values**, **R^2** (explanation).
- Sklearn `LinearRegression().fit(X, y)` → `.coef_`, `.intercept_`, `.score`, `.predict` (prediction).
- Report structure: objective → methods → results → **limitations** → careful conclusion.
- Never: p-value without effect size; causation from correlation; silent extrapolation.

5.12 Practice questions (Unit V)

MCQs

1. `df.describe()` does NOT show:
   a) mean b) 25th percentile c) p-value d) max
2. In a Statsmodels OLS summary, $R^2 = 0.93$ means:
   a) r = 0.93 b) 93% of variation in y explained by the model c) p = 0.07 d) slope = 0.93
3. For sklearn's `.fit(X, y)`, X must be:
   a) 1-D b) 2-D c) A list d) A dictionary
4. The most honest next line after "r = 0.9, p < 0.001" in a report is:
   a) "X causes Y" b) A limitation note c) Nothing d) "p proves causation"

Short questions

5. List the four sections of a statistical report and one line each should contain.
6. Distinguish the roles of Statsmodels and Scikit-learn in one table.
7. Your regression predicts well but n = 12. Write the limitation sentence for the report.

Long / application-based

8. (Illustrative) Using the case dataset, a colleague reports only "slope = 0.35". Rewrite this as a proper three-line results paragraph with r, p-value, R^2, and interpretation.
9. Write full Python code for the describe → correlate → regress workflow on two columns of your choice; explain each line and what the output tells you.
10. A model predicts dissolution = 95 + 0.2 × (mixing time), R^2 = 0.81, slope p = 0.002. A manager wants predictions at triple the max observed mixing time. Write your reply: give the prediction formula's answer, then explain why you refuse to endorse it.

COMBINED REFERENCES — BP201T

1. Lane, D.M., Introduction to Statistics. Houston: Rice University. Available at: https://onlinestatbook.com/2/index.html
2. Walters, S.J., Campbell, M.J. and Machin, D., Medical Statistics: A Textbook for the Health Sciences. Hoboken, NJ: Wiley-Blackwell.
3. Rowe, P., Essential Statistics for the Pharmaceutical Sciences. London: Pharmaceutical Press.
4. Bolton, S. and Bon, C., Pharmaceutical Statistics: Practical and Clinical Applications. New York: Informa Healthcare.
5. De Muth, J.E., Basic Statistics and Pharmaceutical Statistical Applications. Boca Raton, FL: CRC Press.