Assumptions and normal distributions

EXPERIMENTAL DESIGN IN PYTHON

Luke Hayden
Instructor

Summary stats

Mean

- Sum divided by count

Median

- Half of values fall above and below the median

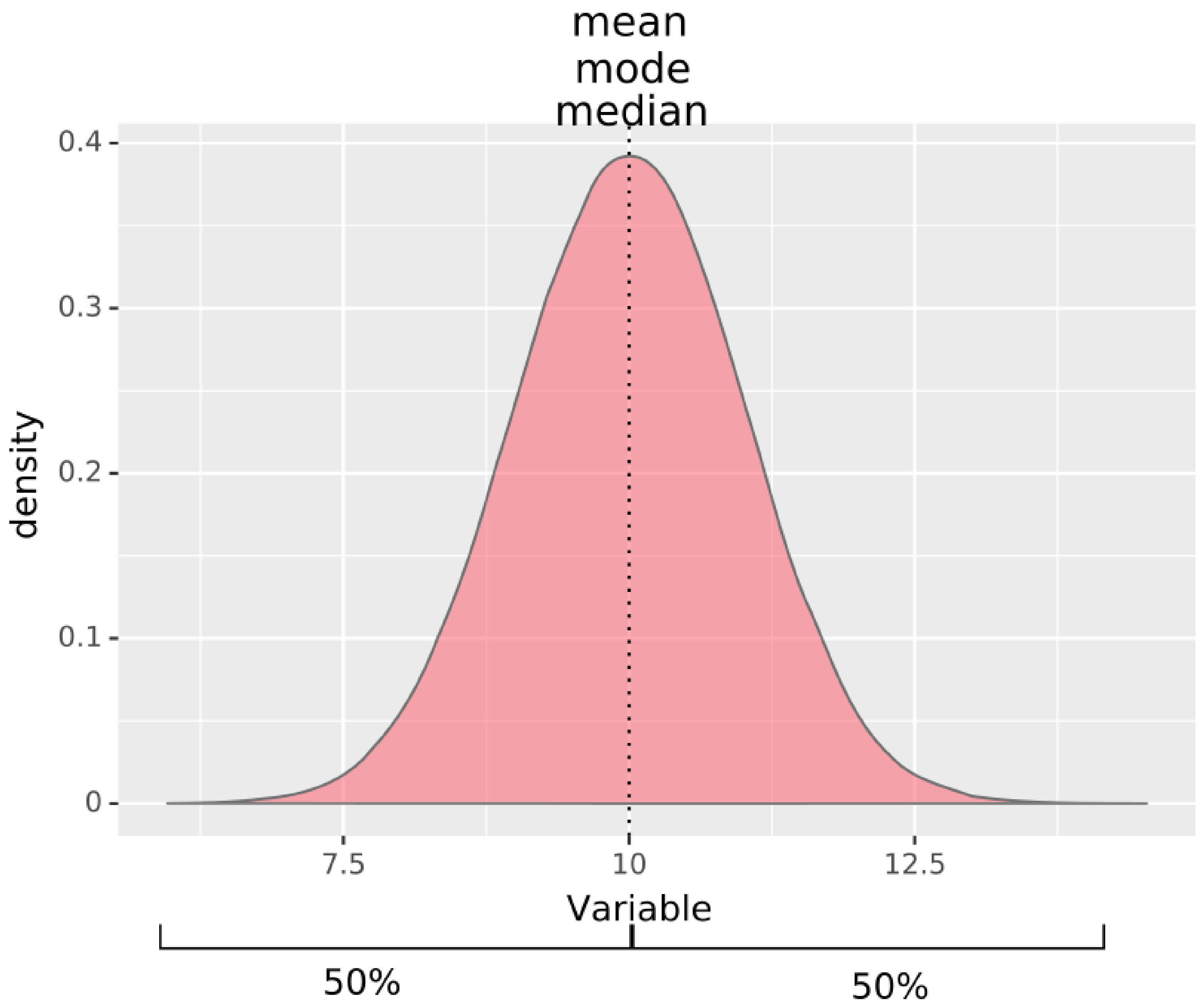
Mode

- Value that occurs most often

Standard deviation

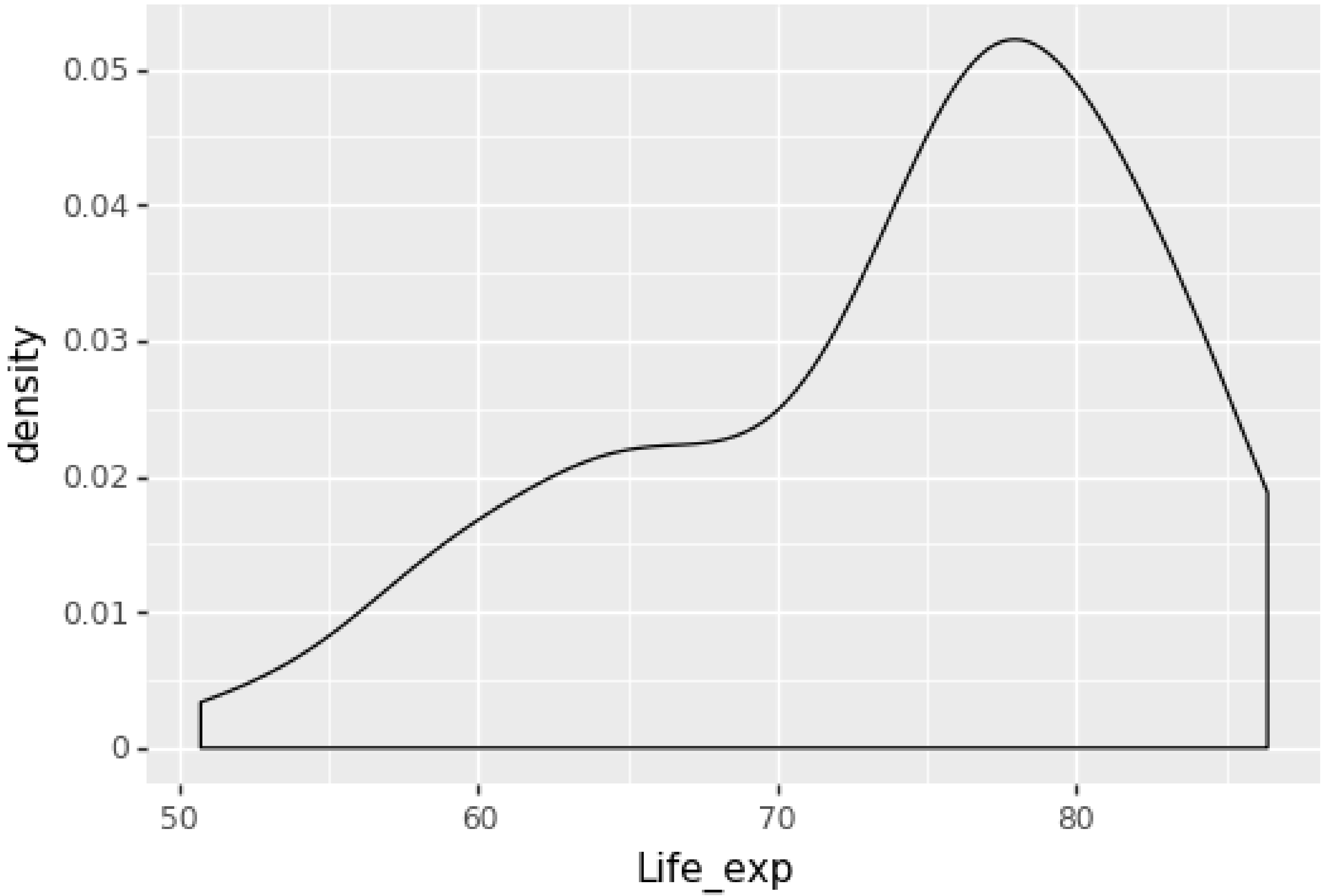
- Measure of variability

Normal distribution



Sample distribution

```
print(p9.ggplot(countrydata)+ p9.aes(x= 'Life_exp')+ p9.geom_density(alpha=0.5))
```



Accessing summary stats

Mean

```
print(countrydata.Life_exp.mean())
```

```
73.68201058201058
```

Median

```
print(countrydata.Life_exp.median())
```

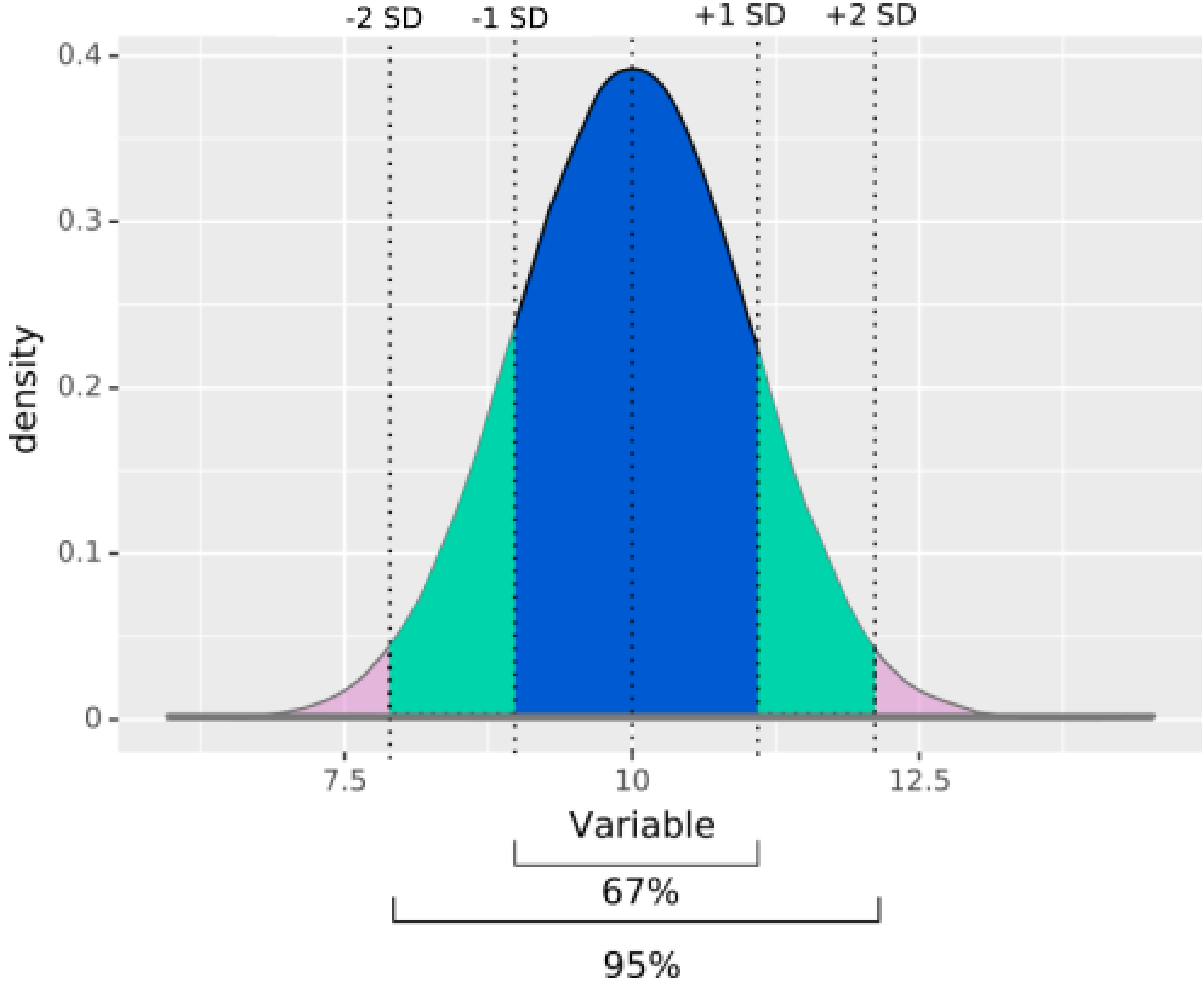
```
76.0
```

Mode

```
print(countrydata.Life_exp.mode())
```

```
78.4
```

Normal distribution



Q-Q (quantile-quantile) plot

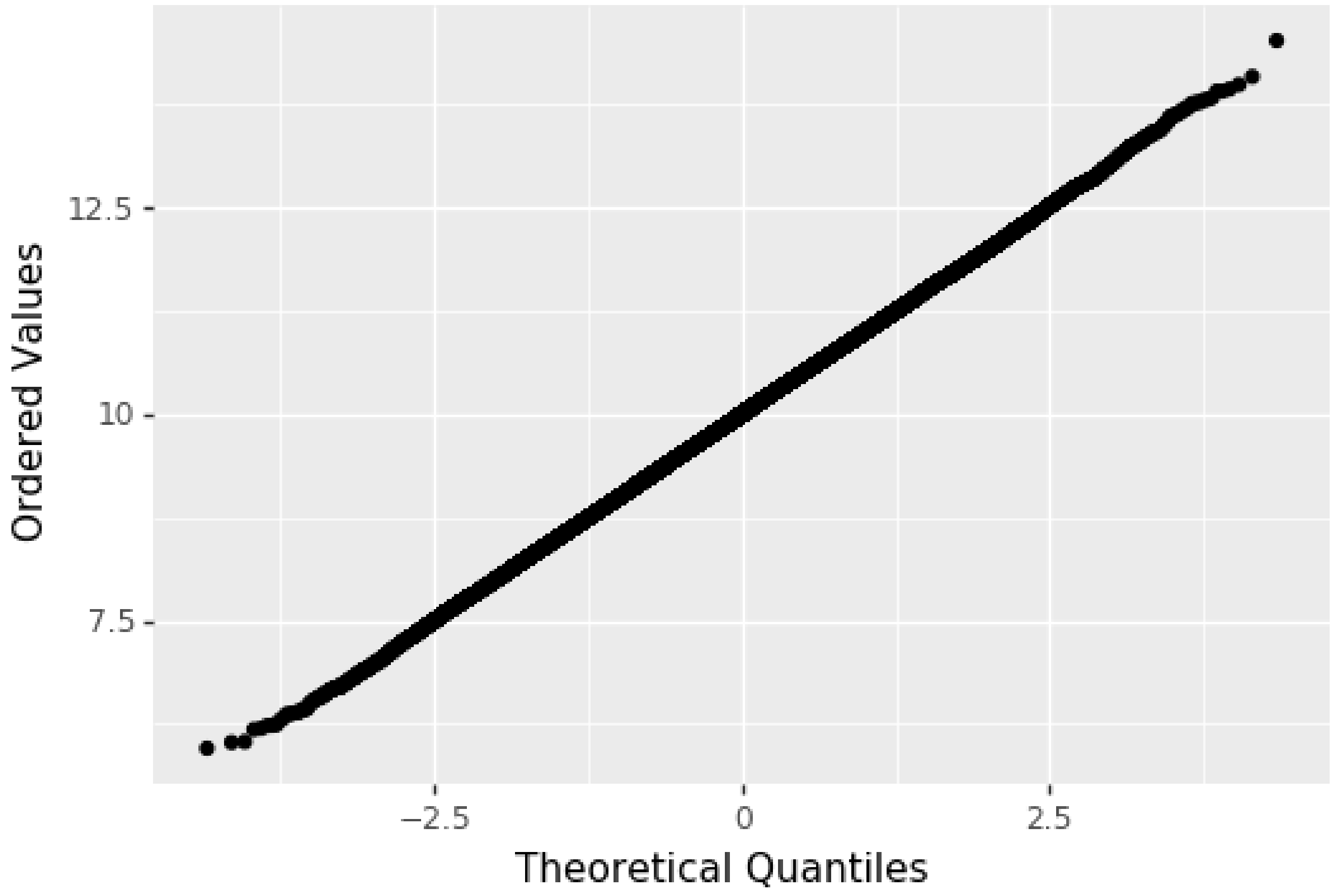
Normal probability plot

Use

- Distribution fit expected (normal) distribution?
- Graphical method to assess normality

Basis

- Compare quantiles of data with theoretical quantiles predicted under distribution

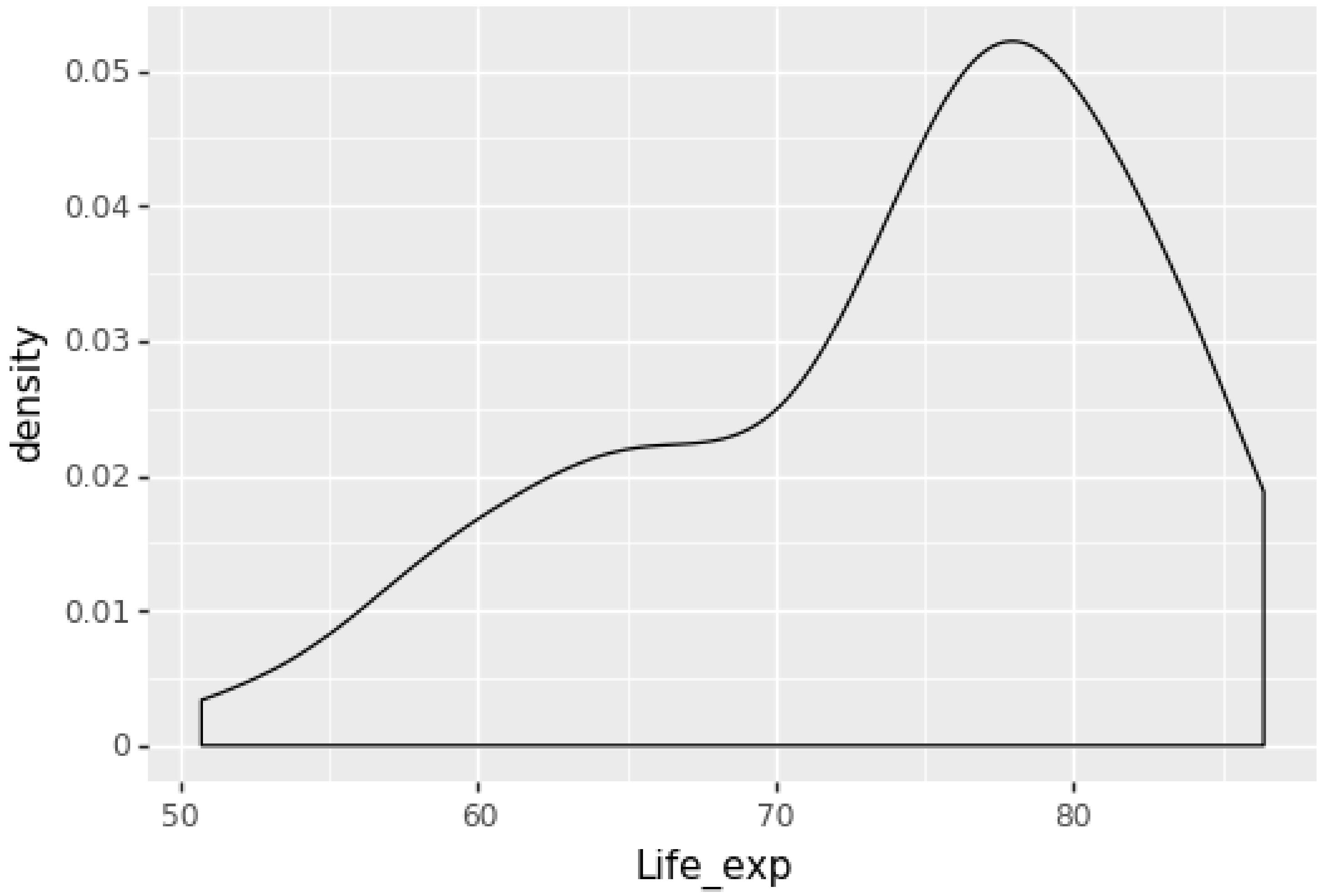


Creating a Q-Q plot

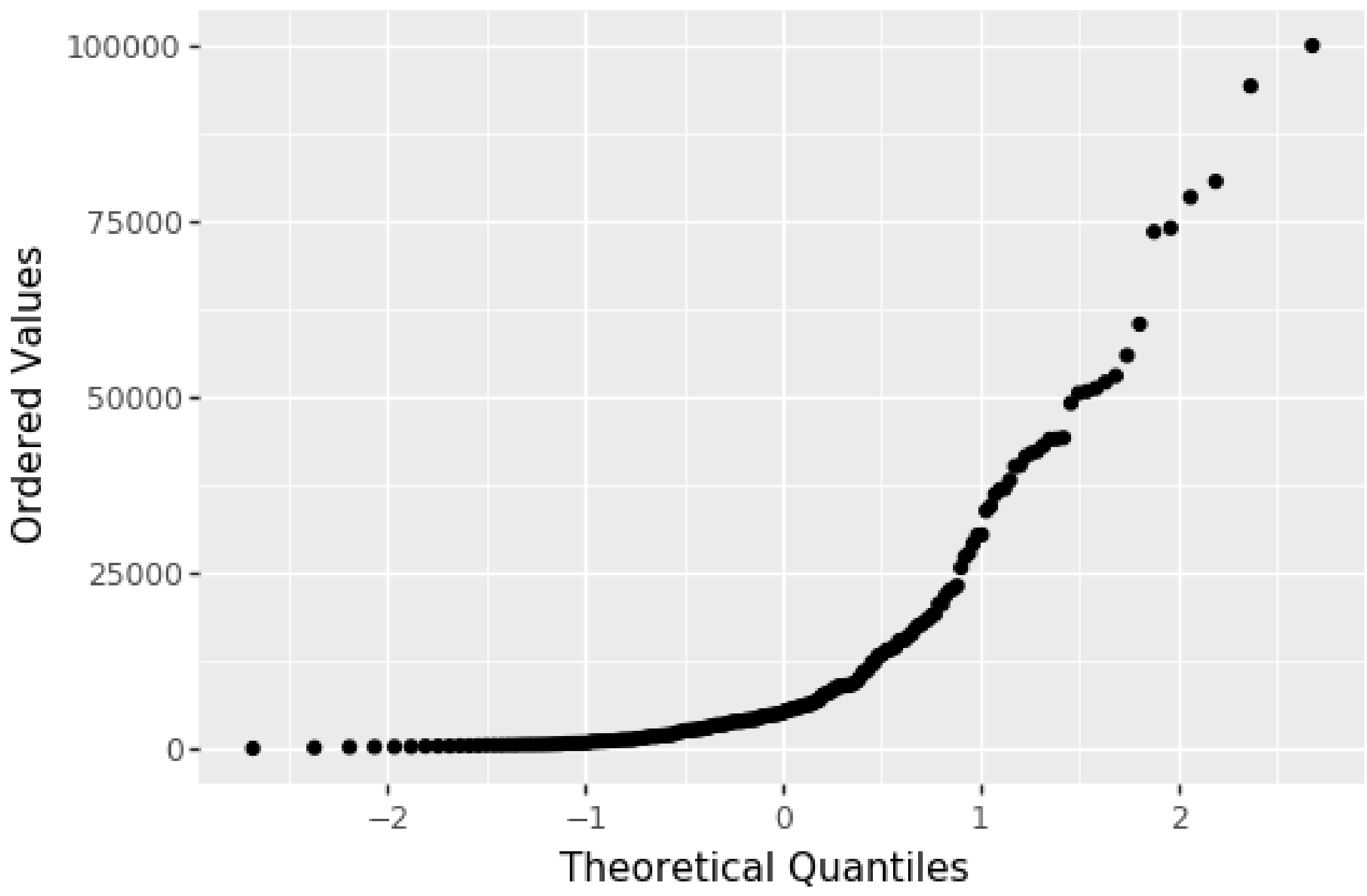
```
from scipy import stats
import plotnine as p9

tq = stats.probplot(countrydata.Life_exp, dist="norm")

df = pd.DataFrame(data = {'Theoretical Quantiles': tq[0][0],
                         "Ordered Values": countrydata.Life_exp.sort_values() })

print(p9.ggplot(df)+ p9.aes('Theoretical Quantiles', "Ordered Values") +p9.geom_point())
```

Q-Q plot for sample

Distribution



Q-Q plot



Let's practice!

EXPERIMENTAL DESIGN IN PYTHON

Testing for normality

EXPERIMENTAL DESIGN IN PYTHON

Luke Hayden
Instructor

Testing for normality

Normal distribution

- Mean, median, and mode are equal
- Symmetrical
- Crucial assumption of certain tests

Approach

- Test for normality

Shapiro-Wilk test

Basis

- Test for normality
- Based on same logic as Q-Q plot

Use

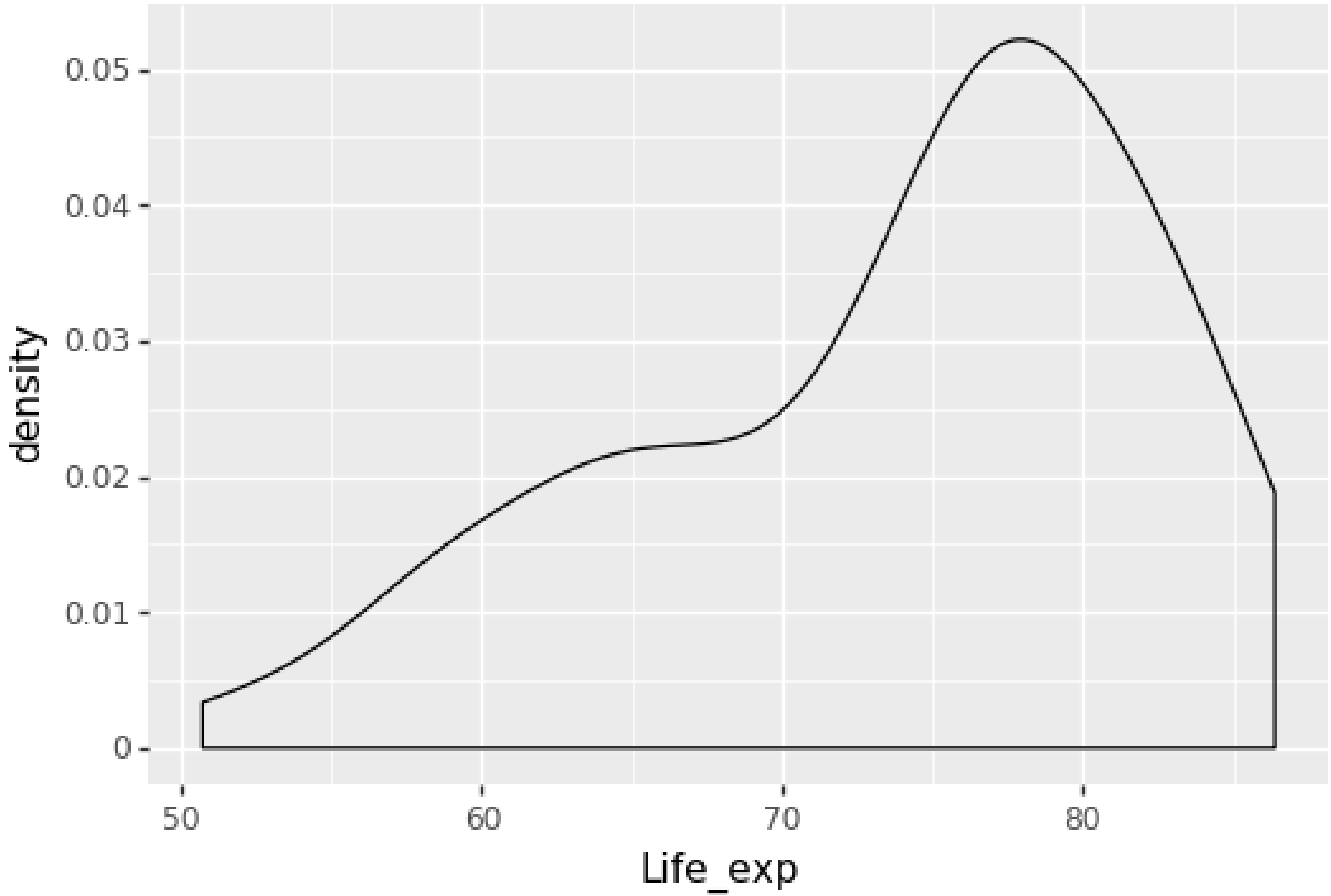
1) Test normality of each sample

2) Choose test/approach

3) Perform hypothesis test

```python
from scipy import stats


shapiro = stats.shapiro(my_sample)
print(shapiro)
```

Shapiro-Wilk test example



Implementing a Shapiro-Wilk test

```
from scipy import stats

shapiro = stats.shapiro(countrydata.Life_exp)
print(shapiro)
```

```
(0.39991819858551025, 6.270842690066813e-26)
```

Test assumptions

Tests based on assumption of normality

- Student's t-test (one and two-sample)
- Paired t-test
- ANOVA

Normality test

- Test by group

Normality and test choice

Sample size & sample mean

- Large sample size: sample mean approaches population mean

Small sample sizes

- Important that normality assumption not violated

Large sample sizes

- Importance of normality is relaxed

Let's practice!

EXPERIMENTAL DESIGN IN PYTHON

Non-parametric tests: Wilcoxon rank-sum test

EXPERIMENTAL DESIGN IN PYTHON

Luke Hayden
Instructor

When assumptions don't hold

- Tests are based on assumptions about data
- Normality: assumption underlying t-test

Violation of assumptions

- Test no longer valid

Approach

- Non-parametric tests
- "Looser" constraints

Parametric vs non-parametric tests

Parametric tests

- Make many assumptions
- Population modeled by distribution with fixed parameters (eg: normal)

Sensitivity

- Higher

Hypotheses

- More specific

Non-parametric tests

- Make few assumptions
- No fixed population parameters
- Used when data doesn't fit these distributions

Sensitivity

- Lower

Hypotheses

- Less specific

Wilcoxon rank-sum vs t-test

Student's t-test

- Parametric

Hypothesis

- mean sample A == mean sample B?

Assumptions

- Relies on normality

Sensitivity

- Higher

Wilcoxon rank-sum test

- Non-parametric

Hypothesis

- random sample A > random sample B

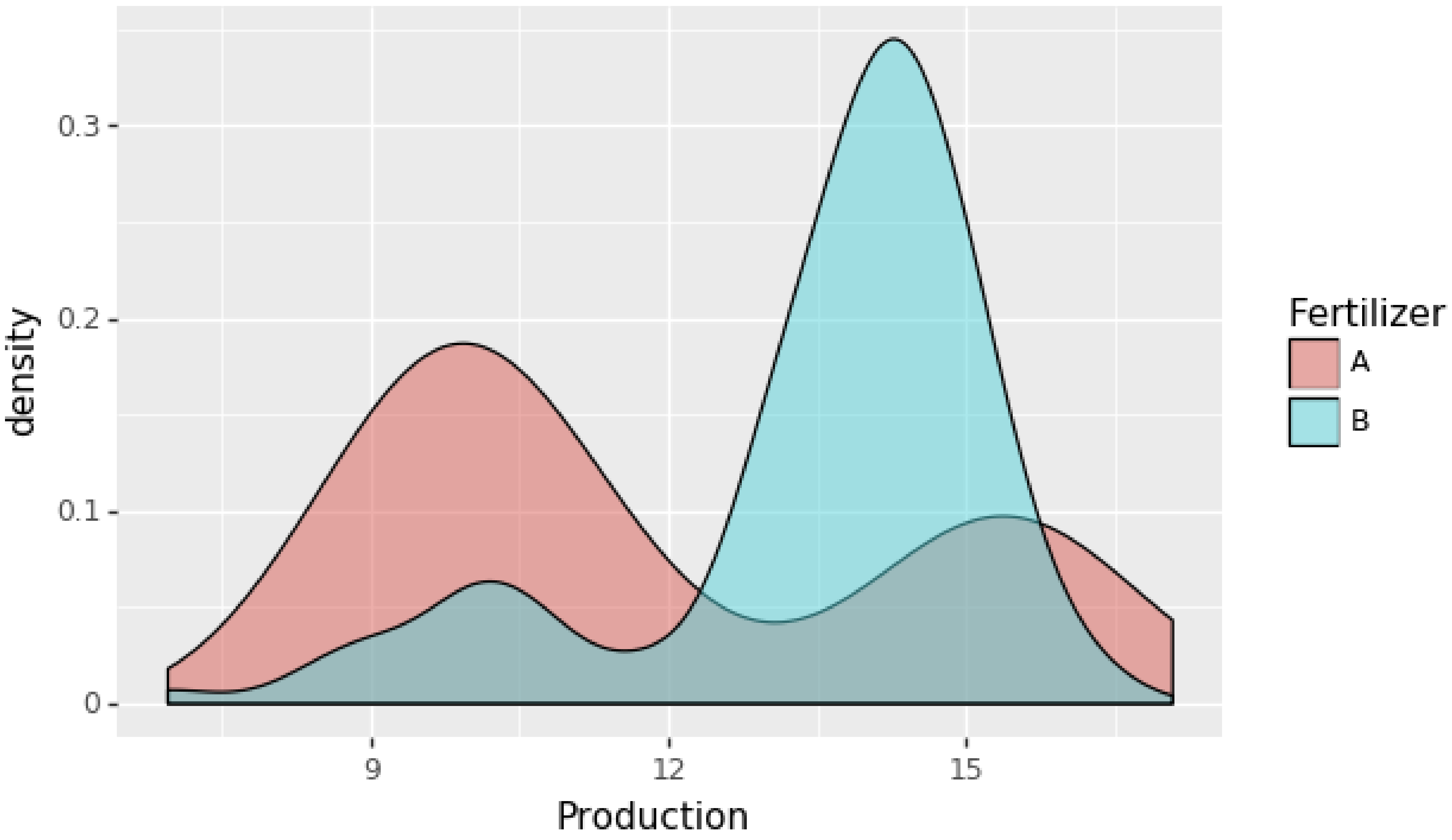
Assumptions

- No sensitive to distribution shape

Sensitivity

- Slightly lower

Wilcoxon rank-sum test example



Implementing a Wilcoxon rank-sum test

```
from scipy import stats

Sample_A = df[df.Fertilizer == "A"]
Sample_B = df[df.Fertilizer == "B"]

wilc = stats.ranksums(Sample_A, Sample_B)
print(wilc)
```

```
RanksumsResult(statistic=16.085203659039184, pvalue=3.239851573227159e-58)
```

Wilcoxon signed-rank test

- Non-parametric equivalent to paired t-test
- Tests if ranks differ across pairs

2017 yield	2018 yield
60.2	63.2
12	15.6
13.8	14.8
91.8	96.7
50	53
45	47

Wilcoxon signed-rank test example

```python
from scipy import stats

yields2018= [60.2, 12, 13.8, 91.8, 50, 45,32, 87.5, 60.1,88 ]
yields2019 = [63.2, 15.6, 14.8, 96.7, 53, 47, 31.3, 89.8, 67.8, 90]

wilcsr = stats.wilcoxon(yields2018, yields2019)
print(wilcsr)
```

```
WilcoxonResult(statistic=1.0, pvalue=0.00683774356850919)
```

Let's practice!

EXPERIMENTAL DESIGN IN PYTHON

More non-parametric tests: Spearman correlation

EXPERIMENTAL DESIGN IN PYTHON

Luke Hayden
Instructor

Correlation

Basis

- Relate one continuous or ordinal variable to another
- Will variation in one predict variation in the other?

Pearson correlation

- Based on a linear model

Pearson vs Spearman correlation

Pearson correlation

- Parametric
- Based on raw values
- Sensitive to outliers

Assumes:

- Linear, monotonic relationship

Effect measure

- Pearson's r

Spearman correlation

- Non-parametric
- Based on ranks
- Robust to outliers

Assumes:

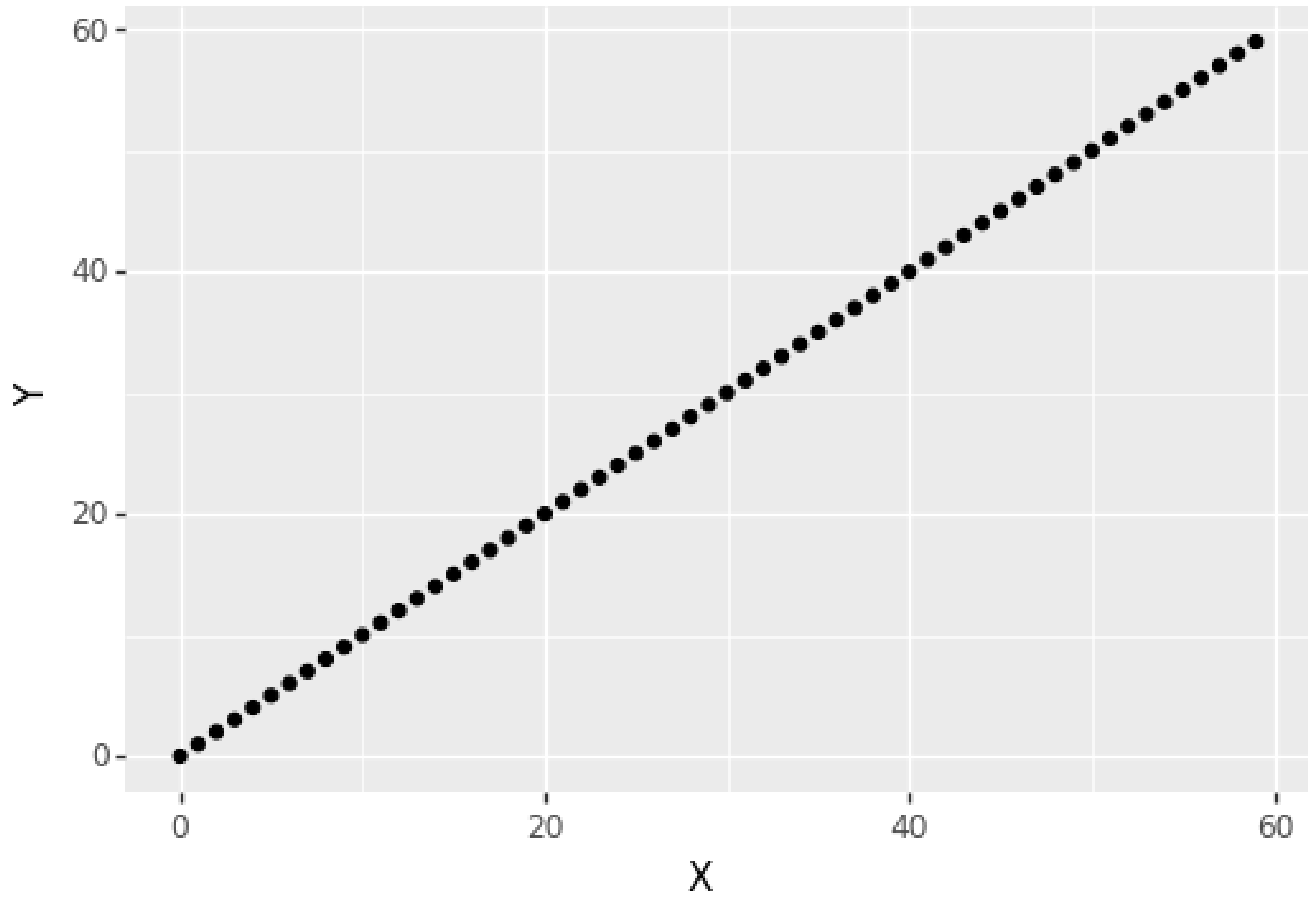
- Monotonic relationship

Effect measure

- Spearman's rho

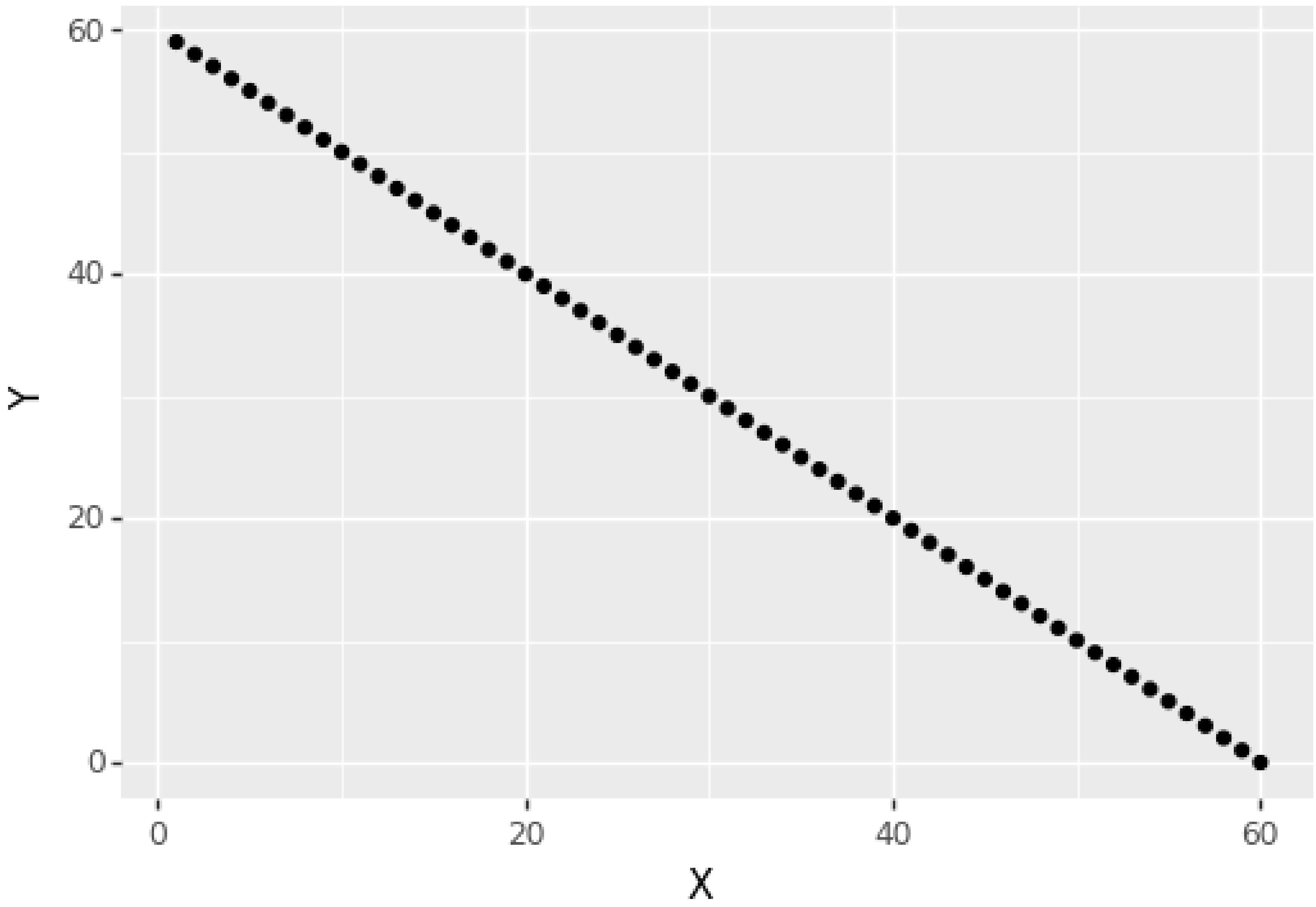
Pearson vs Spearman correlation

Pearson's r: 1, Spearman's rho = 1



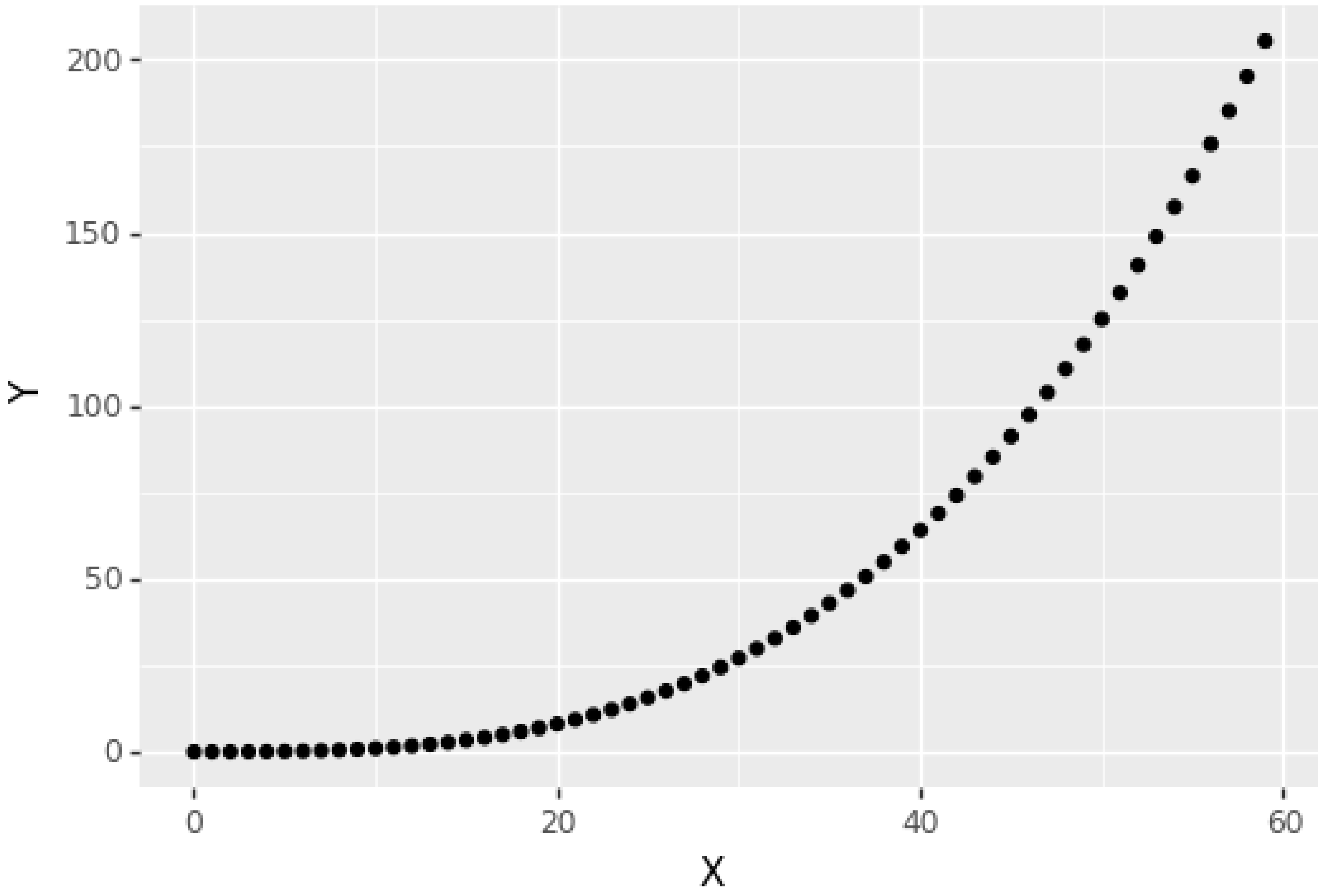
Pearson vs Spearman correlation

Pearson's r: -1, Spearman's rho = -1



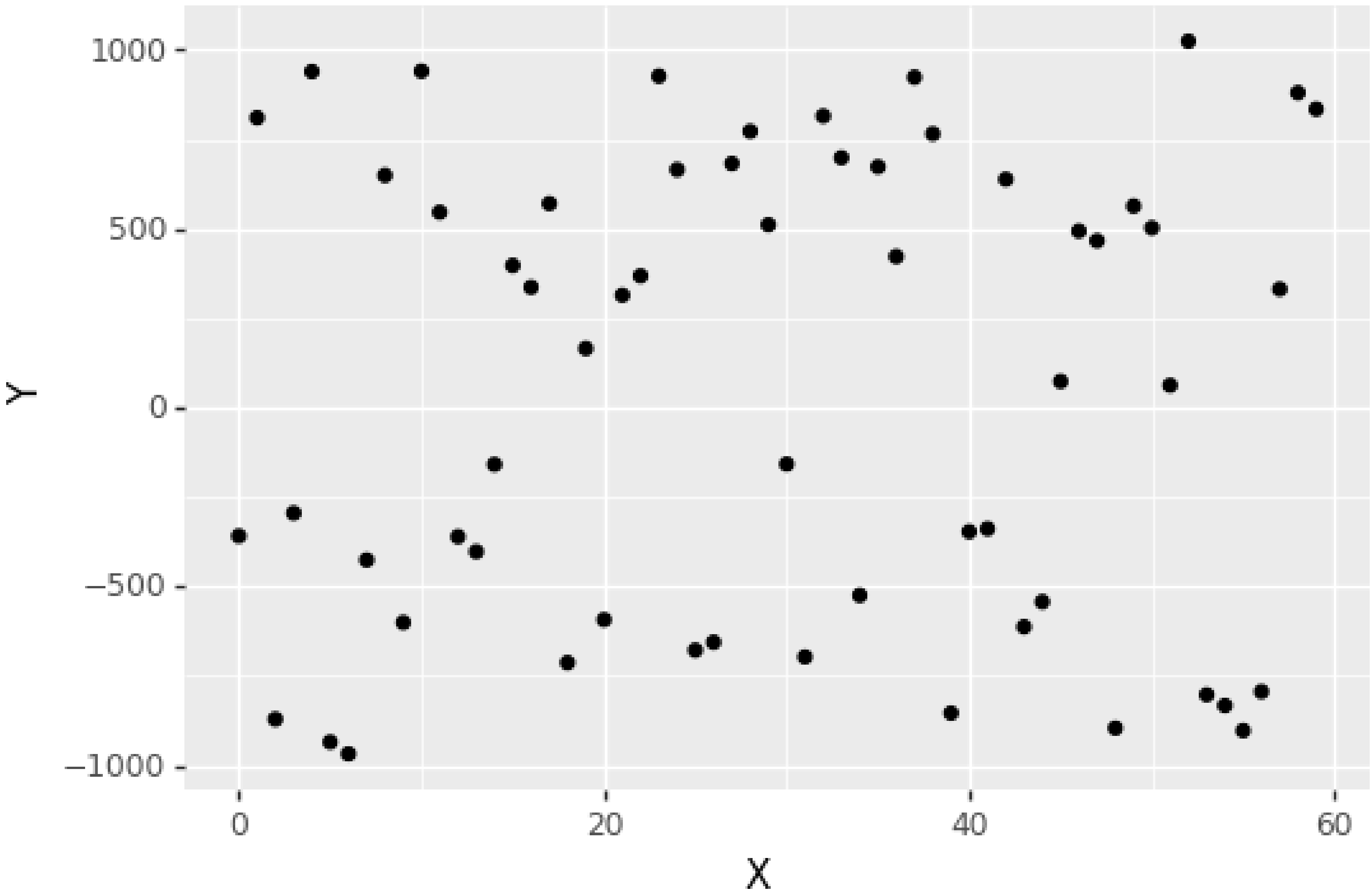
Pearson vs Spearman correlation

Pearson's r: 0.915, Spearman's rho = 1

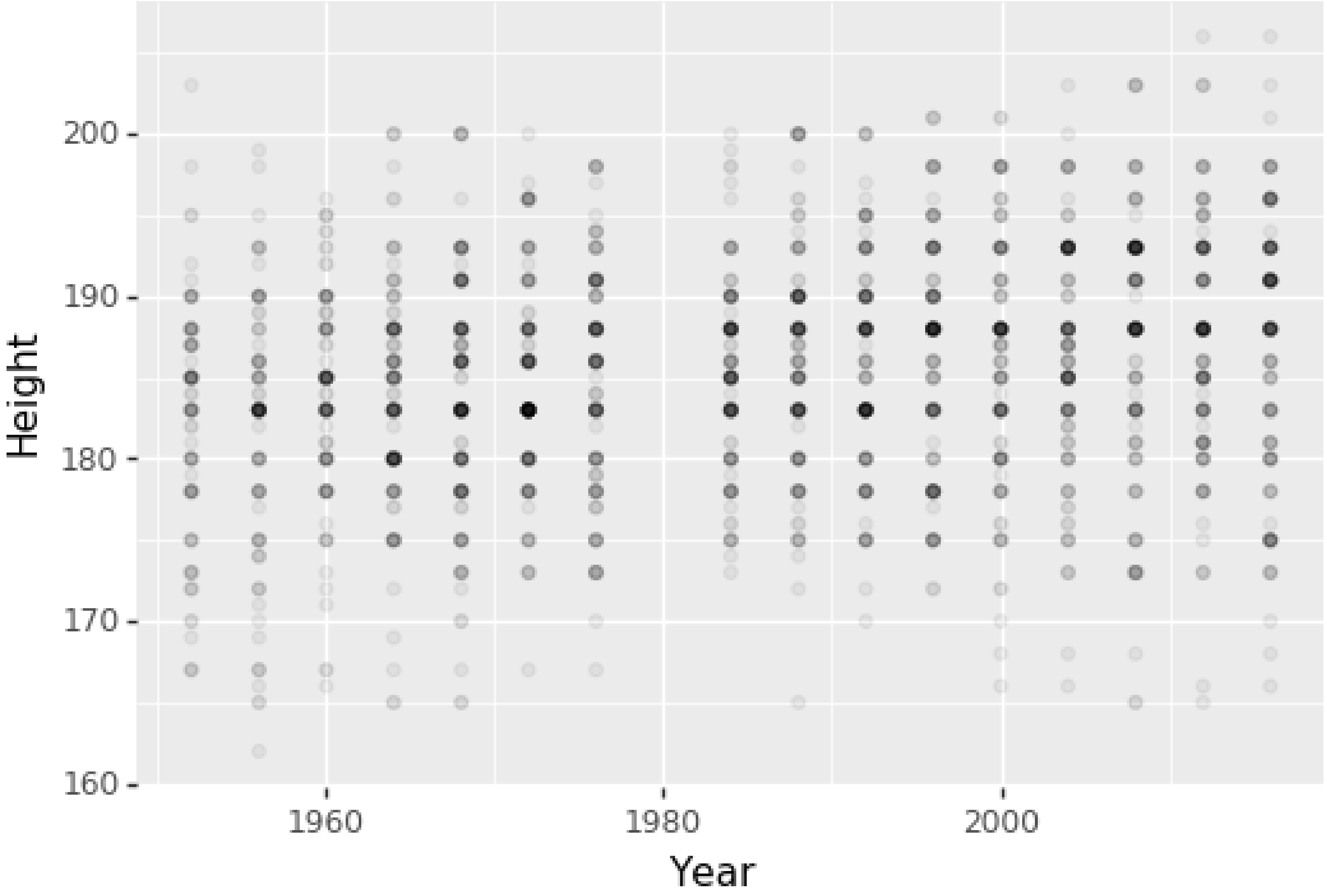


Pearson vs Spearman correlation

Pearson's r: 0.0429, Spearman's rho = 0.0428



Spearman correlation example



Implementing a Spearman correlation

```
from scipy import stats
pearcorr = stats.pearsonr(oly.Height, oly.Weight)
print(pearcorr)
```

```
(0.6125605419882442, 7.0956520885987905e-190)
```

```
spearcorr = stats.spearmanr(oly.Height, oly.Weight)
print(spearcorr)
```

```
SpearmanrResult(correlation=0.728877815423366, pvalue=1.4307959767478955e-304)
```

Let's practice!

EXPERIMENTAL DESIGN IN PYTHON

Summary

EXPERIMENTAL DESIGN IN PYTHON

Luke Hayden
Instructor

What you've learned

Chapter 1

- Exploratory data analysis & hypothesis testing

Chapter 2

- Dealing with multiple factors

Chapter 3

- Type I and II errors and the power-sample size-effect size relationship

Chapter 4

- Dealing with assumptions of tests

Uncertainty is a theme of statistics

Uncertainty is always present

- We can't expect absolute certainty

Approach

- Quantify our uncertainty
- Assess likelihood of competing hypotheses
- Methods may rest on unproven assumptions

Embrace uncertainty!

EXPERIMENTAL DESIGN IN PYTHON