Synthesizing insights from complex experiments

EXPERIMENTAL DESIGN IN PYTHON

James Chapman
Curriculum Manager, DataCamp

datacamp

Manufacturing yield data

```
manufacturing_yield
```

BatchID	MaterialType	ProductionSpeed	TemperatureSetting	YieldStrength
39	Polymer	Medium	Optimal	58.83
195	Metal	High	High	51.29
462	Polymer	High	Optimal	55.15
696	Composite	Medium	Low	50.27
142	Composite	High	Low	57.62

- Multifactorial design: `MaterialType`, `ProductionSpeed`, `TemperatureSetting`
- Response variable: `YieldStrength`

Manufacturing quality data

```
manufacturing_quality
```

```
 BatchID ProductionSpeed  ProductQuality
     149             Low           93.87
     739            High           93.35
     617          Medium           90.45
     131            High           90.26
     684             Low           91.62
```

- Design: `ProductionSpeed`
- Response variable: `ProductQuality`

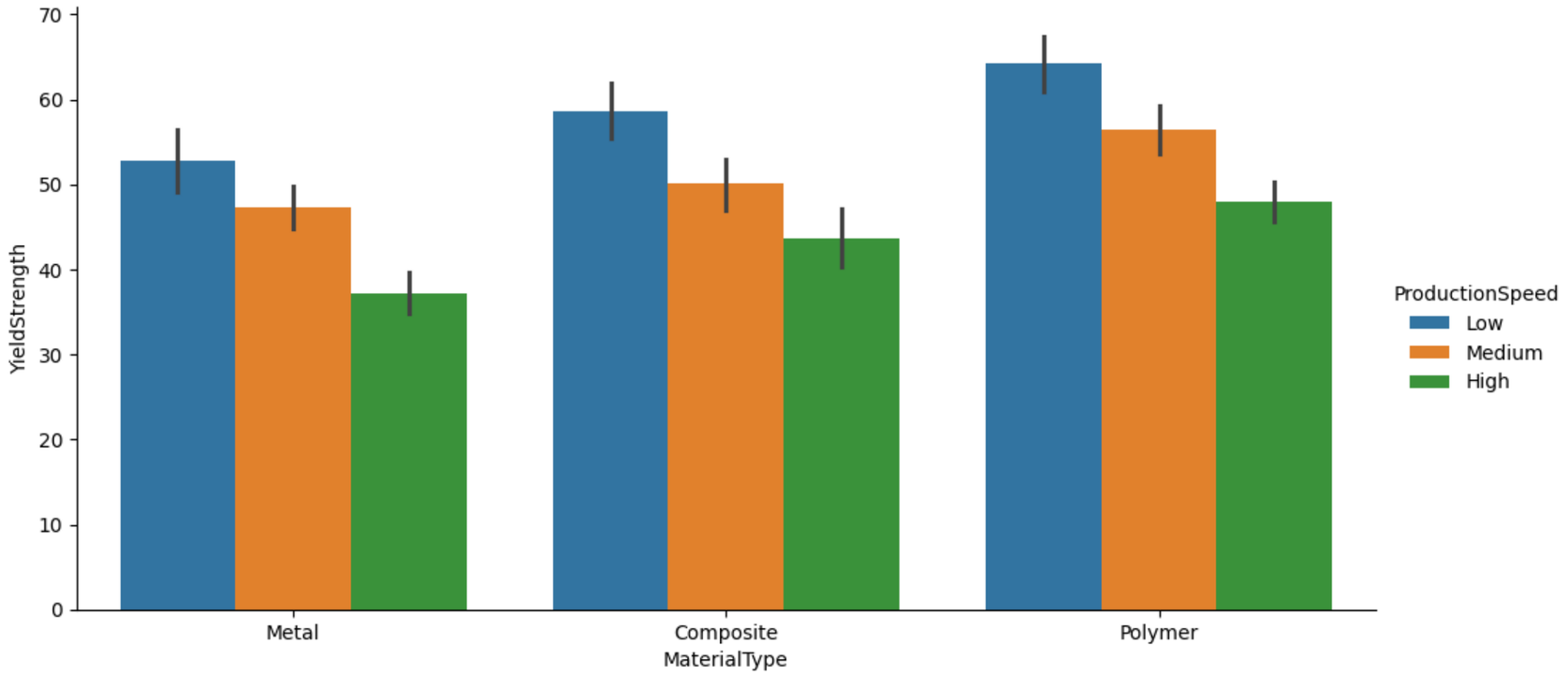
Merging strategy

```
merged_manufacturing = pd.merge(manufacturing_yield,
                        manufacturing_quality,
                        on=['BatchID', 'ProductionSpeed'])
print(merged_manufacturing)
```

BatchID	MaterialType	ProductionSpeed	TemperatureSetting	YieldStrength	ProductQuality
1	Metal	Low	High	57.32	91.19
5	Composite	Medium	Optimal	51.82	90.20
7	Polymer	Low	High	56.12	91.66
8	Composite	High	Optimal	50.91	93.05
11	Polymer	Low	High	50.13	92.31

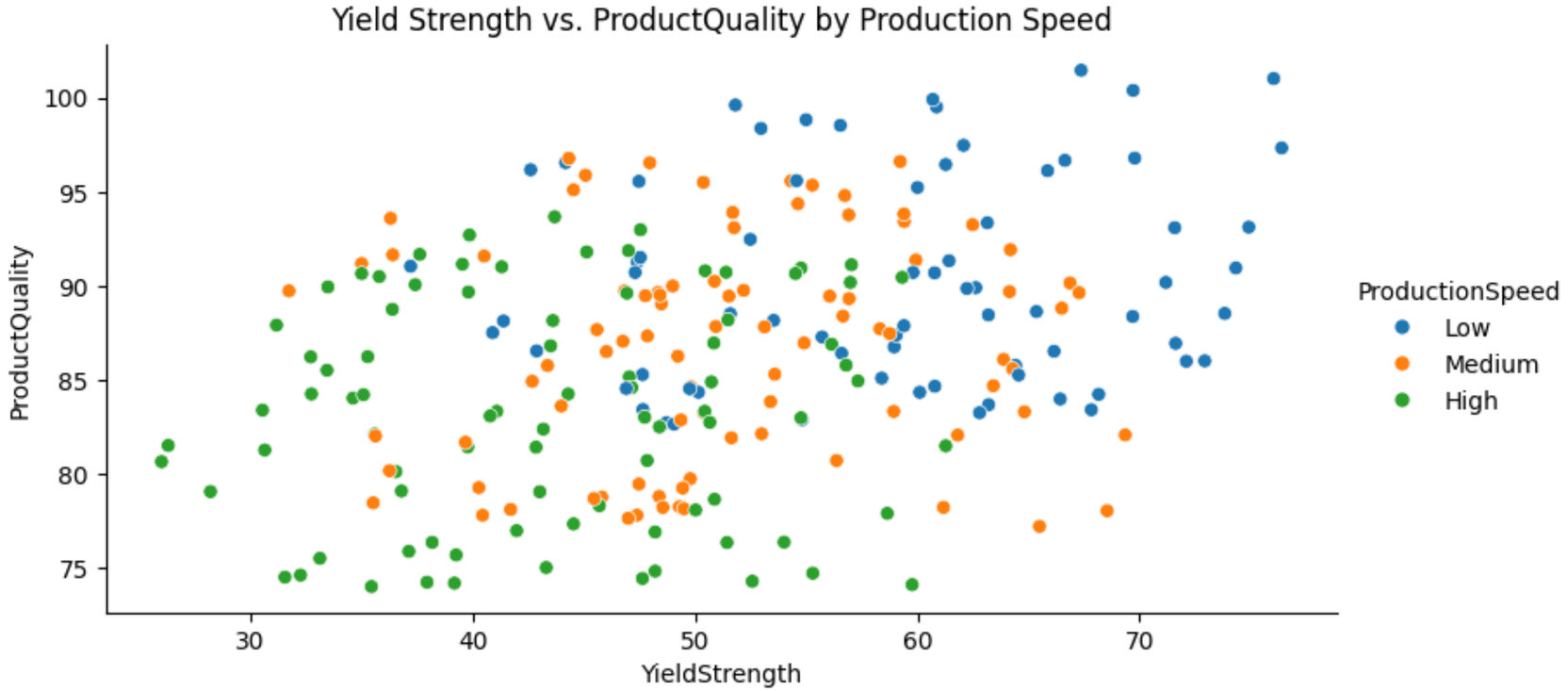
Side-by-side bar graph

```
import seaborn as sns
sns.catplot(x='MaterialType', y='YieldStrength', hue='ProductionSpeed', kind='bar',
            data=merged_manufacturing)
```



Three variable scatterplot

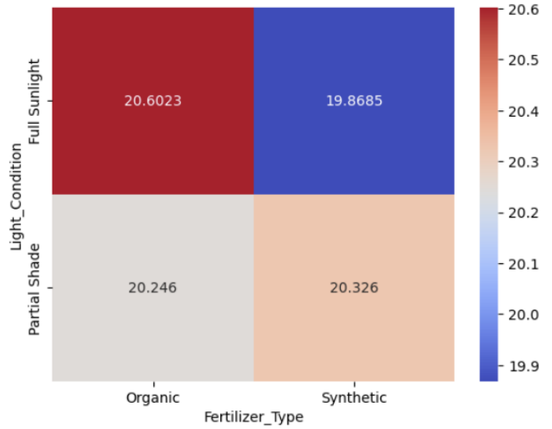
```
sns.relplot(x='YieldStrength', y='ProductQuality', hue='ProductionSpeed',
            kind='scatter', data=merged_manufacturing)
plt.title('Yield Strength vs. Product Quality by Production Speed')
```



Communicating data to technical audiences

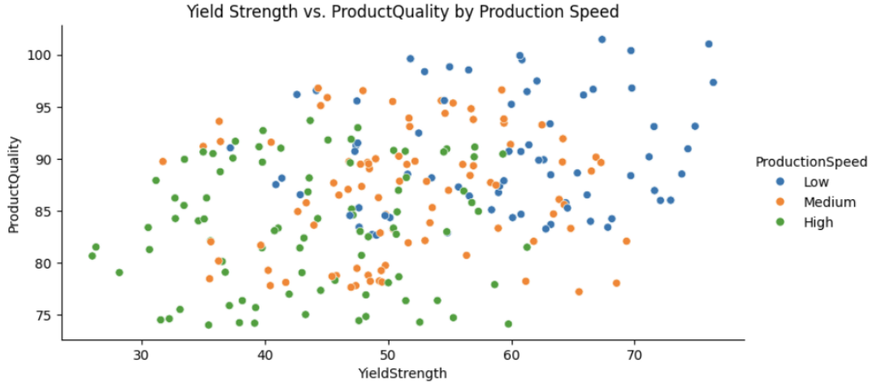
- Craft **data narratives**
  - p-values
  - Test statistics
  - Significance levels
- **Visualize** complex data
  - Heat maps
  - Scatter plots with multiple colors
  - Projections

"The test statistic was **7.78**"



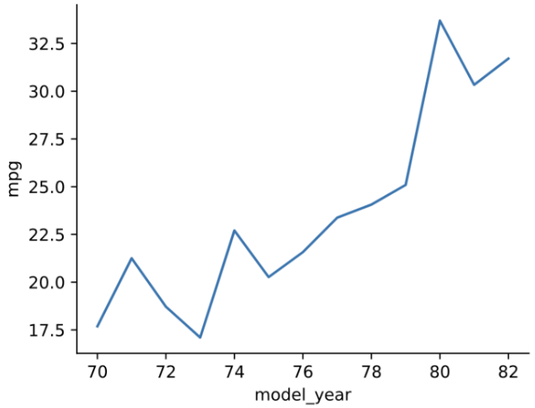
"The p-value is **0.023**"

"We opted for a **5%** significance level"



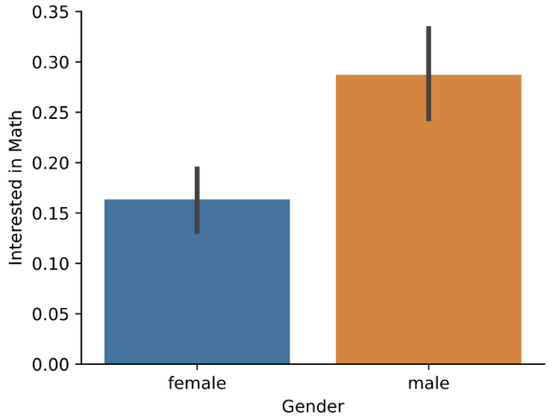
Engaging non-technical audiences with data

- **Simplify** data insights
  - Foundational visualizations: bar and line plots
- Prepare **audience-centric** presentations
  - Why does the data matter?
  - Connect insights to real-world application



"The Ceramic catalyst **significantly increases** rate of reaction"

"The mobile app converts **50% more** customers than the web browser"



Let's practice!

EXPERIMENTAL DESIGN IN PYTHON

Addressing complexities in experimental data

EXPERIMENTAL DESIGN IN PYTHON

James Chapman
Curriculum Manager, DataCamp

datacamp

Geological data

```
mineral_rocks
```

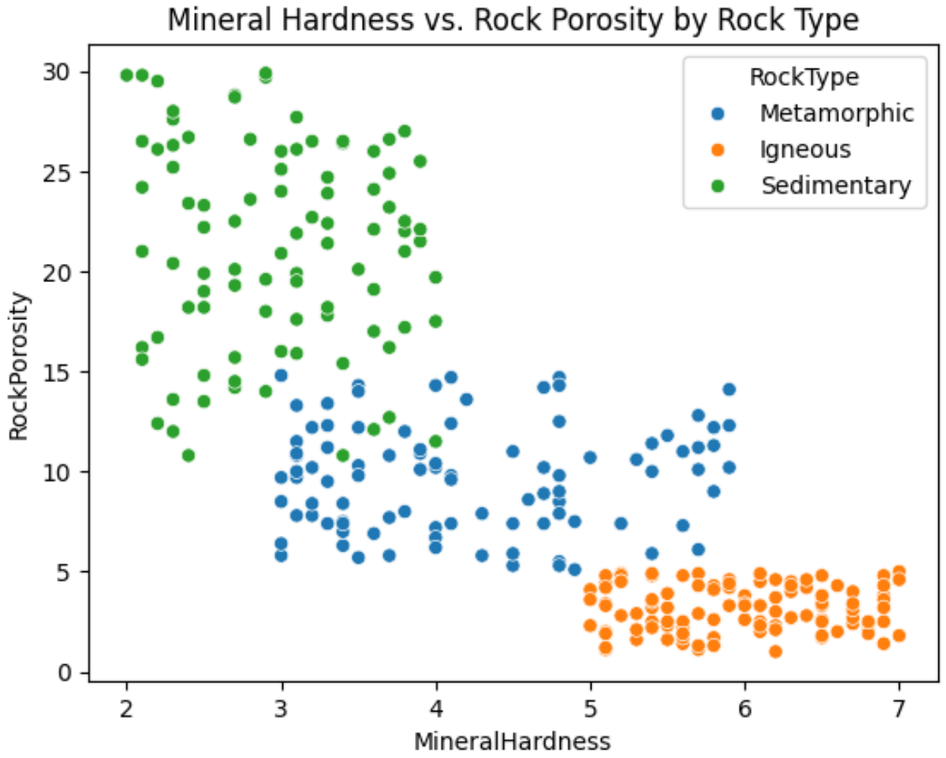
SampleID	RockType	Location	MineralHardness	RockPorosity
1	Metamorphic	West	5.9	12.3
2	Igneous	North	5.3	1.6
3	Metamorphic	East	5.6	11.0
4	Metamorphic	South	3.2	12.2
5	Sedimentary	South	2.0	29.8

Understanding data complexities

- Rock types and mineral hardness may **interact**, impacting mineral properties
- Rock porosity variance may vary, indicating **heteroscedasticity**
- **Confounding** variables could influence mineral hardness and porosity

Addressing interactions

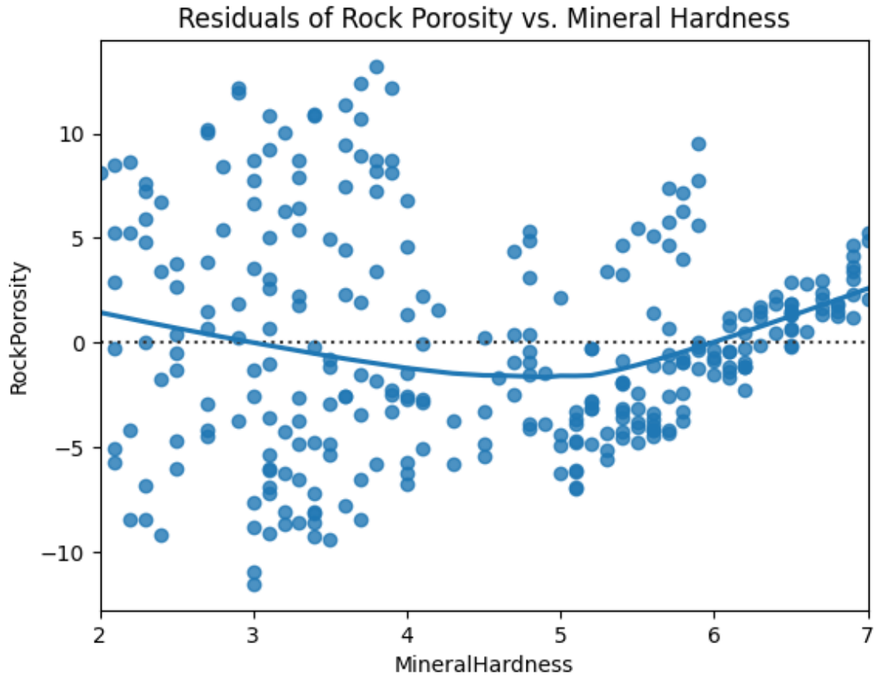
```
sns.scatterplot(x='MineralHardness', y='RockPorosity',
                hue='RockType', data=mineral_rocks)
```



Addressing heteroscedasticity

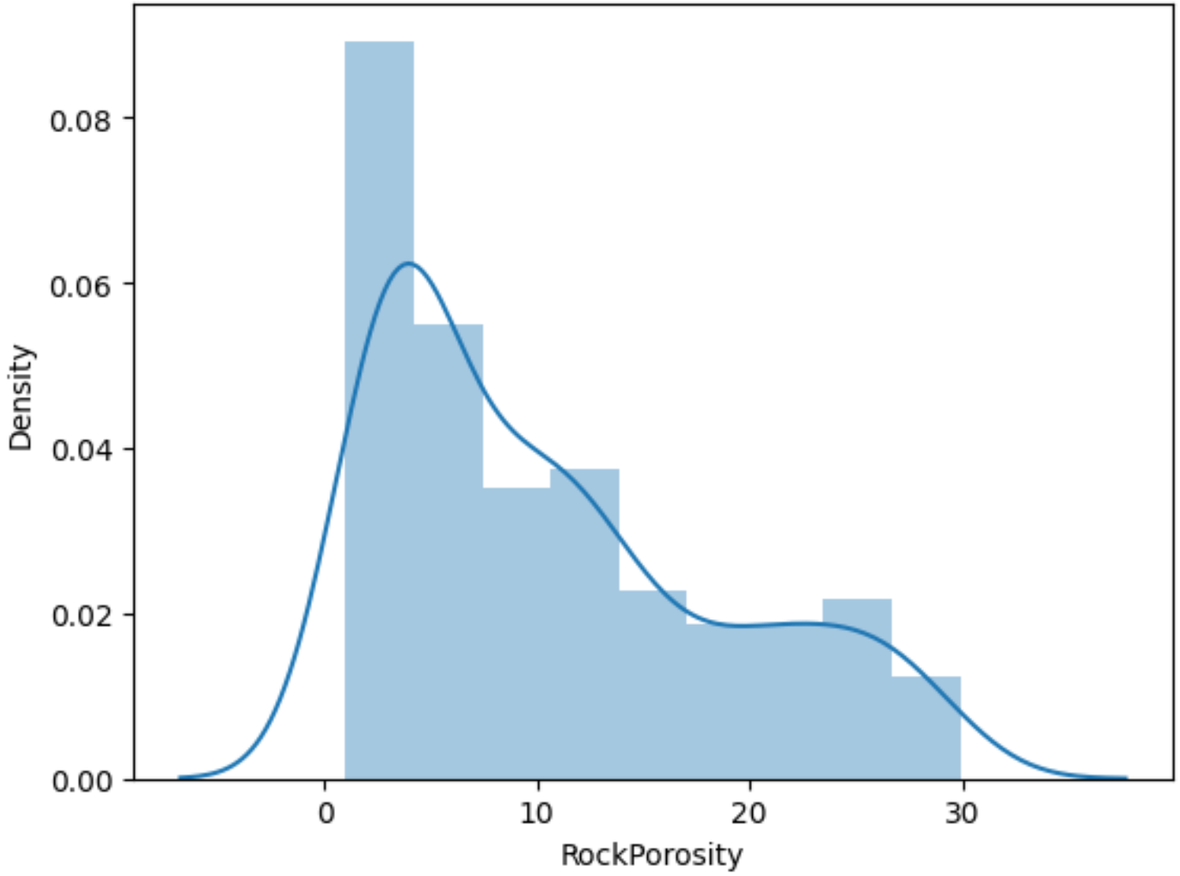
- **Heteroscedasticity:** changing *variability* of a variable across the range of another variable

```
sns.residplot(x='MineralHardness', y='RockPorosity',
              data=mineral_rocks, lowess=True)
```



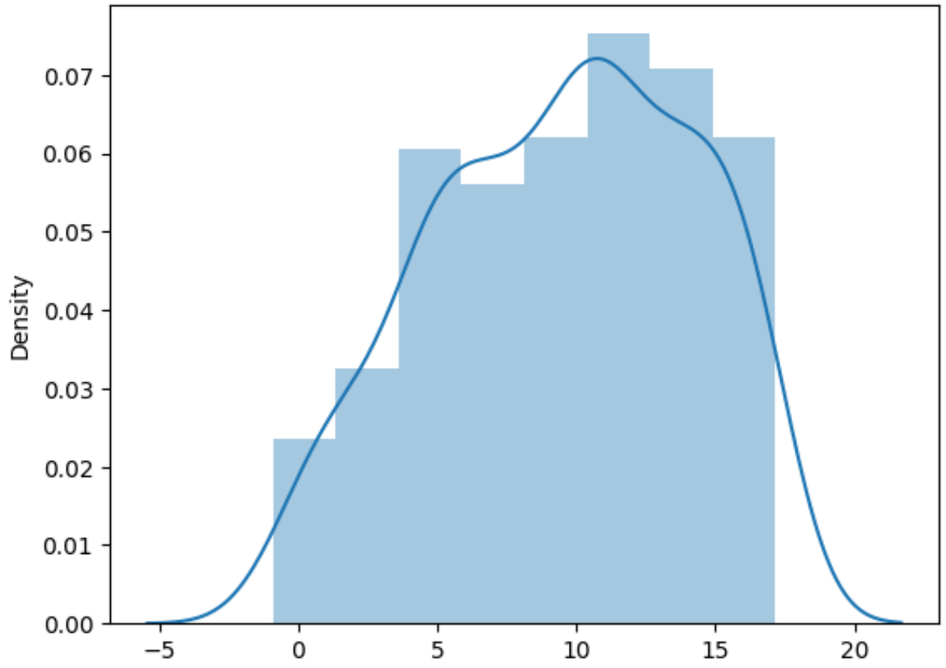
Non-normal data

```
sns.displot(mineral_rocks['RockPorosity'])
```

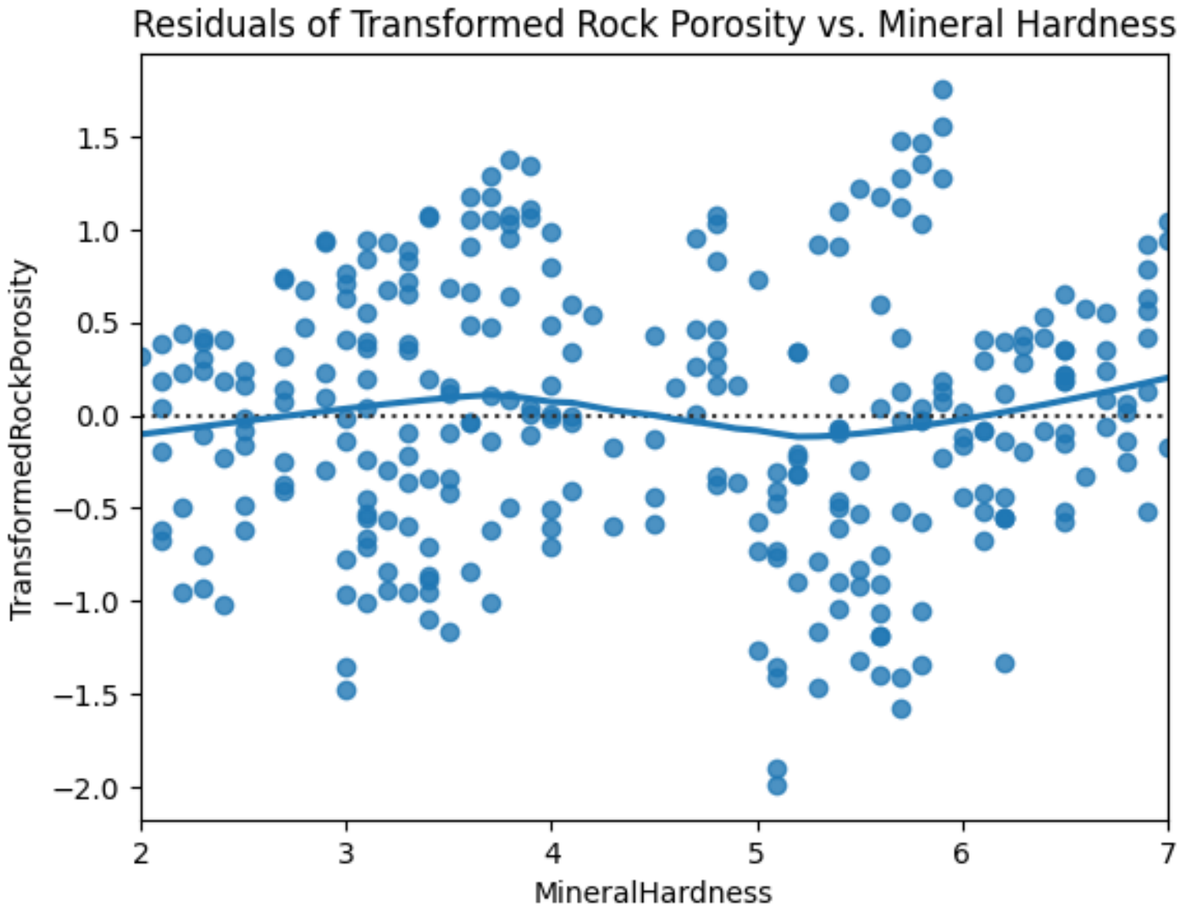


Data transformation with Box-Cox

```
from scipy.stats import boxcox
mineral_rocks['TransformedRockPorosity'], _ = boxcox(mineral_rocks['RockPorosity'])
sns.displot(mineral_rocks['TransformedRockPorosity'])
```



```
sns.residplot(x='MineralHardness', y='TransformedRockPorosity',
              data=mineral_rocks, lowess=True)
```



Let's practice!

EXPERIMENTAL DESIGN IN PYTHON

Applying nonparametric tests in experimental analysis

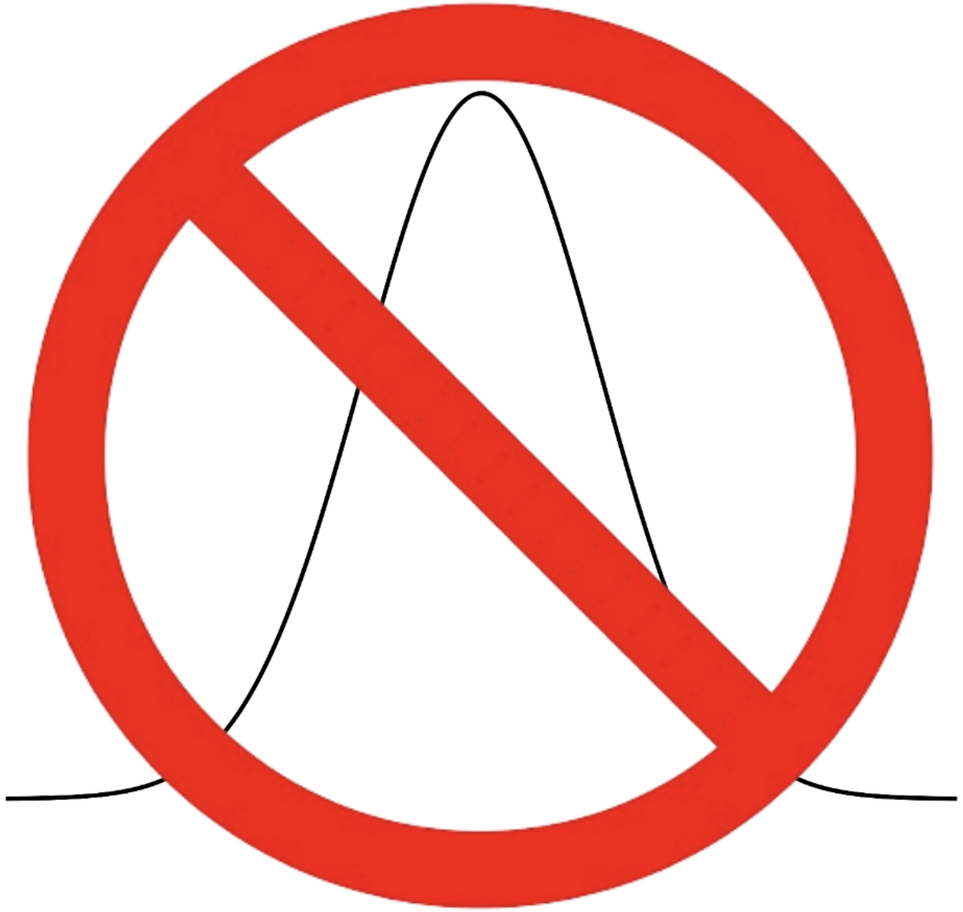
EXPERIMENTAL DESIGN IN PYTHON

James Chapman
Curriculum Manager, DataCamp

datacamp

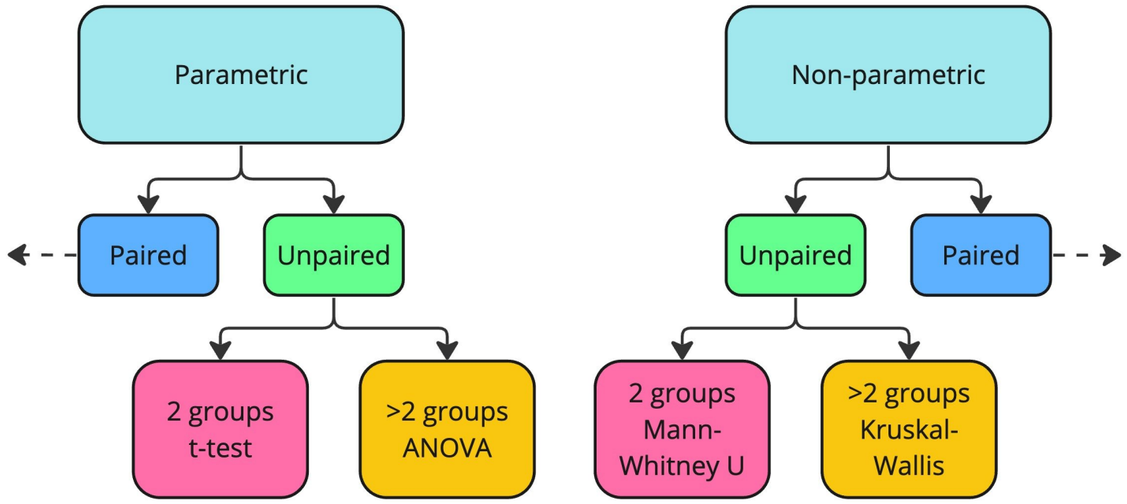
When to use nonparametric tests

- **Parametric test assumptions not met**
- Data on *ordinal scale* or *non-normal*
- Robust to outliers and non-linear data



Exploring nonparametric methods

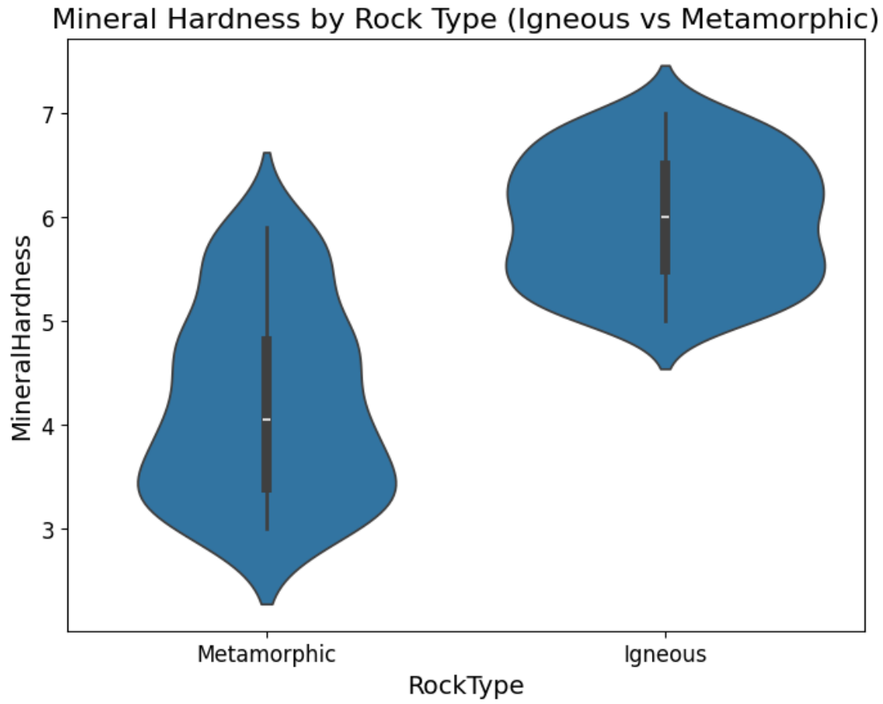
- Nonparametric methods for data that doesn't fit traditional assumptions
- **Mann-Whitney U Test:** compare *two independent* groups
- **Kruskal-Wallis Test:** compare *more than two* groups



Visualizing nonparametric data

- **Violin plots:** Visualizing distributions across groups

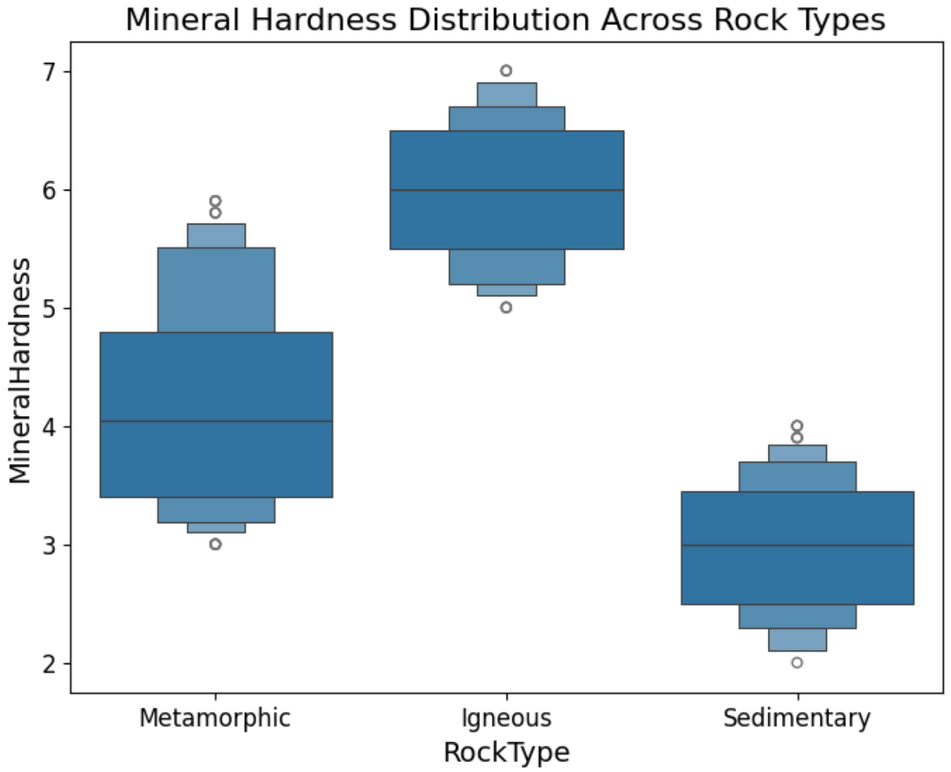
```
condensed_data = mineral_rocks[mineral_rocks['RockType'].isin(['Igneous', 'Metamorphic'])]
sns.violinplot(x='RockType', y='MineralHardness', data=condensed_data)
```



Visualizing nonparametric data

- **Boxen plot**: better insight into distribution shape

```
sns.boxenplot(x='RockType', y='MineralHardness', data=mineral_rocks)
```



Applying nonparametric tests - Mann Whitney U

```
from scipy.stats import mannwhitneyu, kruskal
u_stat, u_pval = mannwhitneyu(
    mineral_rocks[mineral_rocks['RockType'] == 'Igneous']['MineralHardness'],
    mineral_rocks[mineral_rocks['RockType'] == 'Sedimentary']['MineralHardness']
)

print(f"Mann-Whitney U test p-value: {u_pval:.4f}")
```

```
Mann-Whitney U test p-value: 0.9724
```

Applying nonparametric tests - Kruskal-Wallis

```
k_stat, k_pval = kruskal(
    mineral_rocks[mineral_rocks['RockType'] == 'Igneous']['MineralHardness'],
    mineral_rocks[mineral_rocks['RockType'] == 'Sedimentary']['MineralHardness'],
    mineral_rocks[mineral_rocks['RockType'] == 'Metamorphic']['MineralHardness']
)

print(f"Kruskal-Wallis test p-value: {k_pval:.4f}")
```

```
Kruskal-Wallis test p-value: 0.0630
```

Let's practice!

EXPERIMENTAL DESIGN IN PYTHON

Congratulations!

EXPERIMENTAL DESIGN IN PYTHON

James Chapman
Curriculum Manager, DataCamp

datacamp

Chapter 1

- Experiment and data setup
- Normal data

Chapter 2

- Experiment and data setup
- Normal data
- **Factorial and randomized block designs**
- **Covariate adjustment**

Chapter 3

- Experiment and data setup
- Normal data
- Factorial and randomized block designs
- Covariate adjustment
- **Picking the right hypothesis test**
- **Post-hoc analysis**
- **P-values, alpha, and test errors**
- **Power analysis**

Chapter 4

- Experiment and data setup
- Normal data
- Factorial and randomized block designs
- Covariate adjustment
- Picking the right hypothesis test
- Post-hoc analysis
- P-values, alpha, and test errors
- **Data storytelling with experiments**
- **Transformations and nonparametric tests**

What next?

Real-world projects:

- Hypothesis Testing with Men's and Women's Soccer Matches
- Hypothesis Testing in Healthcare
- Experimental Design Project

Applied statistics courses:

- A/B Testing in Python
- Analyzing Survey Data in Python
- Bayesian Data Analysis in Python
- Foundations of Inference in Python

Congratulations!

EXPERIMENTAL DESIGN IN PYTHON