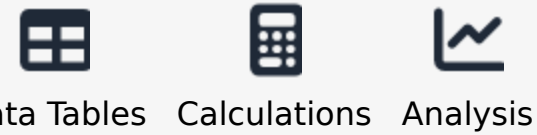
Python for Data Science: NumPy and Pandas



Practical Examples of Powerful Tools for Making Big Data Easy and Efficient Across Multiple Domains

From Biology to Business: Simple Approaches for Non-Technical Users

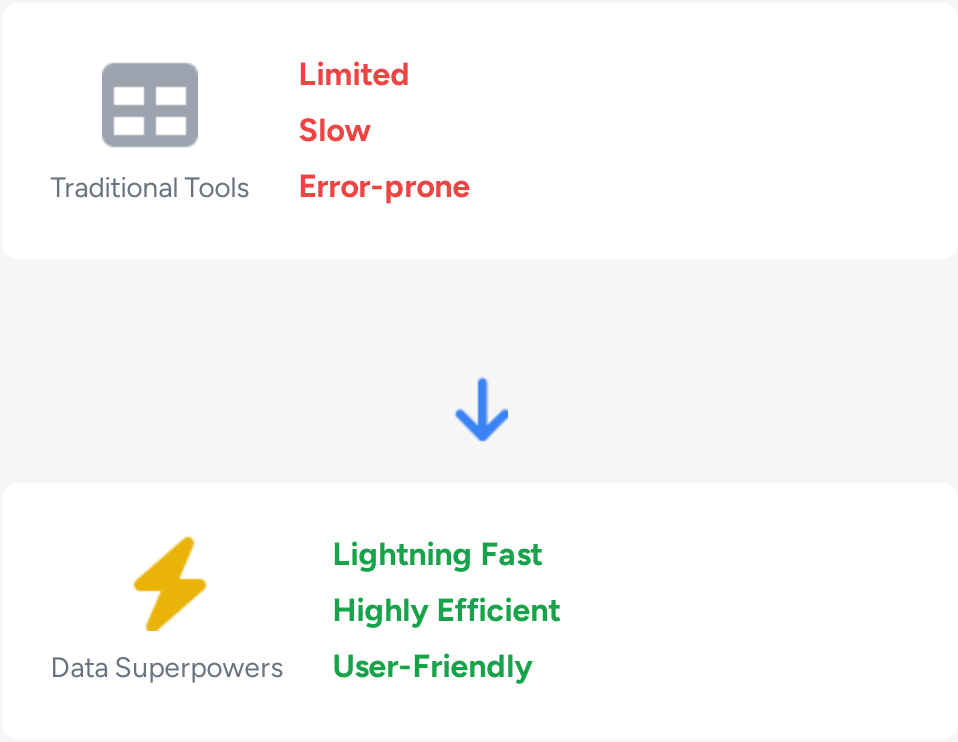
October 7, 2025

Why We Need Data Superpowers

The Data Challenge: We're swimming in data—from patient records to sales figures—and traditional tools can't keep up.

- Spreadsheets become slow and crash with millions of rows
- Manual calculations take too much time
- Complex analyses lead to errors

Our Super Tools: NumPy and Pandas transform raw data into meaningful insights with incredible speed and efficiency.



NumPy: The Super-Fast Calculator

NumPy is a specialized library for Python that introduces a new type of data container called an **ndarray** (N-dimensional array).

Think of it as a highly organized, multi-shelf bookcase where every shelf holds items of the exact same type—like a shelf just for numbers, or a shelf just for text.

This strict organization allows NumPy to work with these items much more efficiently than a regular Python list, which can hold a

How NumPy Stores Data

NumPy stores data in a contiguous block of memory

Much faster than Python lists!

⚡ Lightning-Speed Calculations

NumPy performs operations on entire arrays at once, making calculations incredibly fast compared to looping through lists in pure Python.

Efficient Memory Usage

Data is stored in a contiguous block of memory, allowing for efficient access and manipulation of large datasets without the overhead of Python objects.

Vectorized Operations

Mathematical operations can be applied to entire arrays with a single line of code, eliminating the need for explicit loops and making code more readable.

NumPy in Action: Simple Examples

Array Creation

Creating an array of numbers from 0 to 9

```
import numpy as np

my_array = np.arange(10)
my_array
```

```
[0 1 2 3 4 5 6 7 8 9]
```

💡 Like a super-fast list, but more powerful for calculations

Mathematical Operations

Adding 5 to each element in the array

```
import numpy as np

my_array = np.arange(5)
result = my_array + 5
result
```

```
[5 6 7 8 9]
```

⚡ Broadcasting allows operations across entire arrays instantly

Statistical Functions

Calculating the mean of an array

```
import numpy as np

data = np.array([10, 20, 30, 4
mean = np.mean(data)
mean
```

```
30.0
```

📈 Built-in statistical functions for quick analysis

ℹ️ These simple examples show how NumPy makes numerical operations on large datasets fast and easy

Pandas: The Smart Data Organizer

What is Pandas? A Python library designed for working with datasets, like a "supercharged spreadsheet" or "organized filing cabinet" that can hold different types of information.

Building on NumPy: Pandas uses NumPy's speed for numbers and extends it to structured data with rows and columns.

Key Features:

DataFrame

Tables with labeled rows and columns for different data types

Data Cleaning

Fix missing values and inconsistencies

Filtering & Querying

Find specific information easily

Visualization

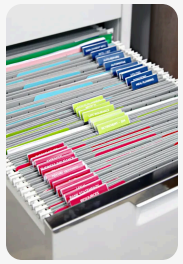
Create charts and graphs from data

How Pandas Organizes Data

Column 1	Column 2	Column 3	Column 4
Row 1	Data	123	Text
Row 2	More	456	Data
Row 3	And	789	Text

"Pandas is like an organized filing cabinet with labeled folders"

The DataFrame: Your Data Workbench



What is a DataFrame? A two-dimensional, table-like structure with labeled rows and columns that can hold different types of information.

Labeled Rows & Columns

Each row and column has a label, making data access intuitive

Multiple Data Types

Different columns can hold numbers, text, dates, and more

Filterable

Easily select specific rows based on conditions

Analyzable

Perform calculations across entire columns with one command

Think of it like: An organized filing cabinet with labeled folders and dividers

Creating a DataFrame

```
import pandas as pd
# Create a simple DataFrame
data = {
  "Product": ["Laptop", "Phone",
  "Tablet"],
  "Sales": [200, 150, 180],
  "Price": [1200, 800, 600]
}
df = pd.DataFrame(data)
```

Basic Operations

```
# Access a column
df['Product']

# Filter rows
df[df['Sales'] > 170]

# Calculate total sales
df['Sales'].sum()
```

DataFrame Example

Product	Sales	Price
Laptop	200	1200
Phone	150	800
Tablet	180	600

Data Cleaning with Pandas

Common Data Cleaning Challenges

- **Missing Values:** Incomplete data that can throw off analyses
- **Duplicates:** Redundant entries that skew results
- **Inconsistent Data:** Different formats causing errors

```python
# Fix missing values
df.fillna(0)
# Remove duplicates
df.drop_duplicates()
```

Before & After

Raw Data

Product	Price	Sales
Laptop	1200	15
Phone	800	25
Tablet	NULL	12
Headphones	150	40
Smartwatch	200	NULL

Cleaned Data

Product	Price	Sales
Laptop	1200	15
Phone	800	25
Smartwatch	200	0

Biology & Healthcare Applications

NumPy and Pandas transform medical data into life-saving insights, accelerating research and improving patient outcomes.

Key Applications

Genetic Analysis

Analyze DNA sequences to understand genetic structures and disease links

Treatment Comparison

Calculate success rates to identify the most effective treatments

Medical Research

Process patient records to understand diseases and develop new treatments

Patient Care

Improve diagnosis and treatment strategies based on data analysis

Treatment Effectiveness Analysis

```
# Calculate treatment success rates
import numpy as np

# Sample data: 1 = success, 0 = failure
treatment_a_results = np.array([1, 0, 1, 1, 0, 1, 0, 1, 1, 0])
treatment_b_results = np.array([1, 1, 0, 1, 0, 1, 1, 1, 0, 1])

# Calculate success rates
success_rate_a = np.mean(treatment_a_results) * 100
success_rate_b = np.mean(treatment_b_results) * 100

# Display results
print(f"Treatment A Success Rate: {success_rate_a:.0f}%") # 70%
print(f"Treatment B Success Rate: {success_rate_b:.0f}%") # 80%

# Insight: Treatment B is more effective
```

Treatment Effectiveness Analysis

Challenge: Cancer researchers need to compare effectiveness of two treatments to determine which is more successful.

Using NumPy to Calculate Success Rates:

```
# 1 = treatment success, 0 = treatment failure
import numpy as np

# Sample data for Treatment A and B
treatment_a_results = np.array([1, 0, 1, 1, 0, 1, 0, 1,
treatment_b_results = np.array([1, 1, 0, 1, 0, 1, 1, 0,

# Calculate success rates
success_rate_a = np.mean(treatment_a_results) * 100
success_rate_b = np.mean(treatment_b_results) * 100

# Display results
print(f"Treatment A Success Rate: {success_rate_a:.0f}%
print(f"Treatment B Success Rate: {success_rate_b:.0f}%
```

Results:

Treatment A:	70%
Treatment B:	**80%**

Insight:

With just a few lines of NumPy code, researchers can quickly determine that Treatment B is 10% more effective than Treatment A.

This insight can lead to recommending the more effective treatment, potentially saving lives.

Key Benefit: NumPy's speed allows quick analysis of large medical datasets to drive better patient care.

Business & Retail Analytics

Businesses use Pandas to transform sales data into actionable insights for inventory and marketing strategies.

Example: Retail Clothing Store Chain

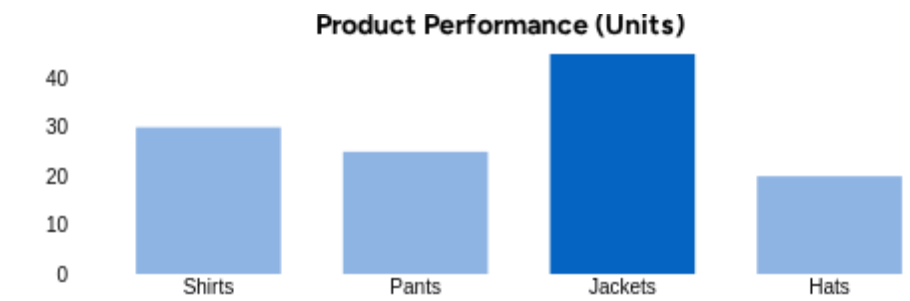
```
import pandas as pd

# Sales data from 50 stores
sales_data = pd.DataFrame({
    'Store': ['NYC', 'LA', 'Chicago', 'Houston'] * 25,
    'Product': ['Shirts', 'Pants', 'Jackets', 'Hats'] * 25,
    'Revenue': [] * 25,
    'Date': pd.date_range(start='1/1/2023', periods=100)
})

# Find best-selling product
best_product = sales_data.groupby('Product')['Revenue'].sum().idx
print(f"Best-selling product: {best_product}")
```

```
Best-selling product: Jackets
```

Sales Analysis Visualization



Inventory Optimization

Stock more of top-selling items and less of underperforming products.

Marketing Decisions

Target promotions based on regional preferences and customer demographics.

Time-Based Analysis

Track seasonal trends and plan staffing based on historical data patterns.

Customer Insights

Analyze customer behavior and preferences across demographics.

Sales Data Analysis Example

Using Pandas to analyze store sales data

Business Challenge

A retail clothing store chain with 50 locations needs to identify their best-selling product to optimize inventory and marketing.

How can they analyze sales data efficiently?

```python
# Import Pandas
import pandas as pd

# Sample sales data
sales_data = pd.DataFrame({
    'Store': ['NYC', 'LA', 'Chicago',
    'Product': ['Shirts', 'Pants', 'Ja
    'Revenue': [] * 25,
    'Date': pd.date_range(start='1/1/2
})

# Find best-selling product
best_product = sales_data.groupby('Pro
print(f"Best-selling product: {best_pr
```

Result

```
Output:
Best-selling product: Jackets
```

Benefits

- Quickly identifies top-performing products
- Optimizes inventory management
- Enables targeted marketing strategies

This analysis could be expanded to include seasonal trends, store-specific performance, and more.

Social Sciences Applications

NumPy and Pandas transform social science data into meaningful insights:

Demographic Analysis

Analyze population trends by age, gender, and geography.

Survey Results

Clean and analyze responses to understand societal behaviors.

Example: Calculating survey response rates by demographic group

```
import pandas as pd

data = {
    'Age Group': ['18-25', '26-35', '36-45', '46-55', '56-65', '66+'],
    'Responses': [150, 280, 320, 210, 180, 120],
    'Population': [200, 350, 400, 300, 250, 150]
}

df = pd.DataFrame(data)
df['Response Rate'] = df['Responses'] / df['Population'] * 100
```

Benefits for Social Scientists

Trend Analysis

Track changes in social behaviors over time.

Pattern Recognition

Identify correlations between different social factors.

Data Cleaning

Handle incomplete or inconsistent survey data.

Educational Research

Analyze learning outcomes across diverse student populations.

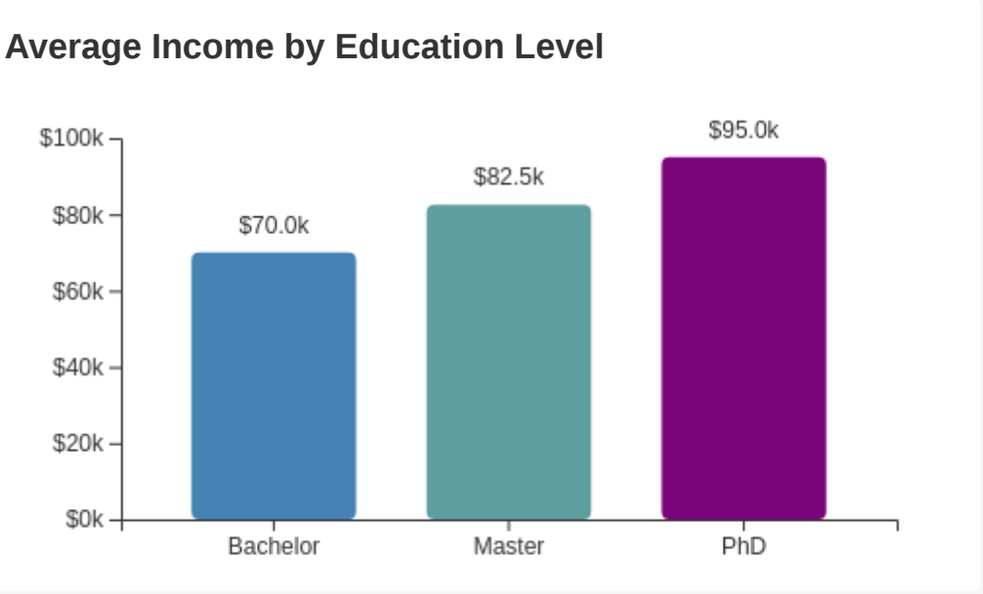
Key Insight: These tools make complex social science data analysis accessible to researchers without coding expertise.

Survey Data Analysis Example

Let's analyze a simple survey dataset to understand how Pandas can help process and visualize social science data.

```
# Load survey data into DataFrame
import
pandas as pd

survey_df = pd.DataFrame({
'Respondent_ID': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],
'Age': [25, 32, 47, 35, 29, 42, 38, 31, 28, 45],
'Gender': ['F', 'M', 'F', 'M', 'F', 'M', 'F', 'F', 'M', 'F'],
'Education': ['Bachelor', 'Master', 'Bachelor', 'PhD', 'Bachelor',
'Master', 'PhD', 'Bachelor', 'Master', 'PhD'],
'Income': [50000, 65000, 80000, 95000, 70000, 85000, 100000,
60000, 75000, 90000]
})
```



Key Takeaway: Pandas makes it easy to clean, group, and analyze survey data to extract meaningful insights about societal trends.

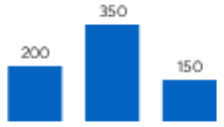
Quick Data Visualization

Creating insightful charts with minimal code

Pandas makes it easy to create visualizations from your data using simple functions.

Simple Bar Chart

```
import pandas as pd
data = {'Product':
['Laptop', 'Phone',
'Tablet'],
```



Pie Chart

```
import pandas as pd
data = {'Category': ['A',
'B', 'C', 'D'],
'Value': [25, 20, 15, 40]}
```

Benefits

Simple Syntax

Quick creation with minimal code

Customizable

Colors, labels, and styles

Integration

Plots directly from DataFrames

Pandas uses Matplotlib for visualizations

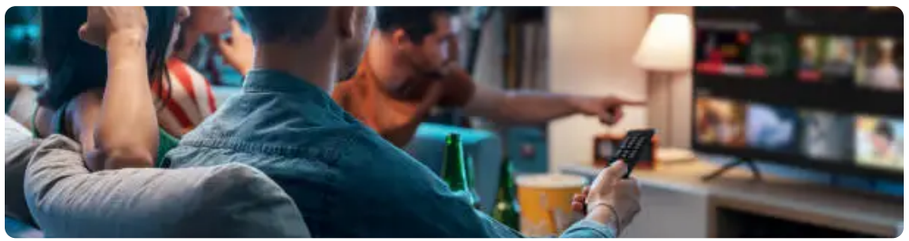
Real-World Impact: Other Domains

Weather Forecasting

NumPy and Pandas analyze thousands of satellite weather images to predict storms and patterns.

Enables processing of massive environmental datasets

Streaming Services



Libraries power recommendation engines suggesting movies based on viewing history.

Handles complex user preference data

Financial Services

Fraud Detection

Detects fraud by identifying unusual patterns in banking data.

Manufacturing

Predictive Maintenance

Optimizes production by analyzing sensor data to prevent breakdowns.

Key Takeaways

The Power of These Libraries

Lightning-Fast Processing

NumPy and Pandas handle large datasets with incredible speed, transforming hours of work into seconds.

Effortless Data Transformation

These libraries convert raw, overwhelming data into clear, actionable insights with minimal code.

Complementary Strengths

NumPy excels at calculations, while Pandas masters organized data—creating a powerful data analysis team.

Why This Matters to You

Appreciate Data-Driven Decisions

Gain insight into the rigorous process behind strategic choices in your field.

Ask Better Questions

Engage more effectively with data scientists, asking informed questions about data processing and conclusions.

Collaborate More Effectively

Contribute meaningfully to data-related projects, bridging the gap between technical and non-technical teams.

Thanks!

Have questions about NumPy, Pandas, or data science applications?

RESOURCES:

NumPy documentation: numpy.org
Pandas documentation: pandas.pydata.org
Python Data Science Handbook by Jake VanderPlas

Python for Data Science: NumPy and Pandas Practical Examples