Block 2: Essential Data Wrangling with NumPy & Pandas

Python Module for Incoming ISE & OR PhD Students

Will Kirschenman

August 7, 2025 | 10:00 AM - 10:50 AM

North Carolina State University

NC STATE UNIVERSITY

Welcome to Block 2!

- **Goal:** Become familiar with the essential tools for data manipulation in Python
- **Duration:** 50 minutes of hands-on data wrangling
- **Format:** Presentation + interactive notebook exercises

What We'll Cover

NumPy arrays · Pandas DataFrames · Real-world data cleaning · PhD dataset analysis

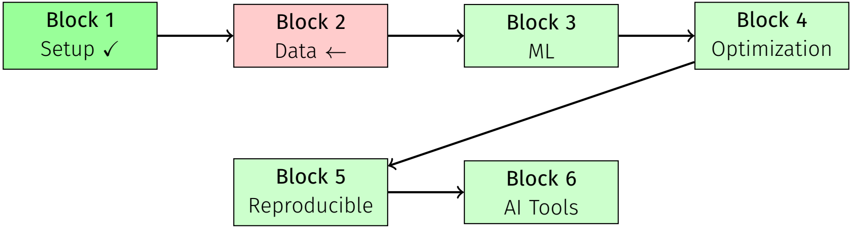
Session Learning Objectives

By the end of Block 2, you will:

1. Understand **NumPy arrays** for efficient numerical computation
2. Master **Pandas DataFrames** for data manipulation and analysis
3. Apply **data cleaning techniques** to messy real-world data
4. Create **new features** from existing data
5. Prepare a **clean dataset** ready for machine learning (Block 3)

Our Mission
Transform a messy PhD student research dataset into analysis-ready data!

Recap: Where We Are



From Block 1

You now know Python basics, Google Colab, and the programming mindset. Time to handle real data!

Our Dataset: PhD Student Research Productivity

What we're working with:

Dataset Features

- **280+ PhD students** at NC State
- **Research productivity** metrics
- **NC State specifics** (Hunt Library, departments)
- **Realistic but messy** data

The Problems

- Missing values
- Duplicate rows
- Inconsistent department names
- Extreme outliers
- Mixed data types

Real Data Reality

This is exactly the kind of messy data you'll encounter in your coursework!

NumPy Fundamentals

Why NumPy? The Performance Story

Python Lists

Slow (interpreted)
Memory inefficient
No vectorization

$\xrightarrow{\text{10-100x faster!}}$

NumPy Arrays

Fast (C backend)
Memory efficient
Vectorized operations

For OR/ISE Coursework
When you’re processing coursework assignments and problem sets, speed matters!

Speed Comparison: Python vs NumPy

Let's see the difference with 1 million data points:

Python List Approach

```python
python_list = list(range(1000000))

# Multiply each element by 2
start_time = time.time()
result = [x * 2 for x in python_list]
python_time = time.time() - start_time

# Result: ~0.15 seconds
```

NumPy Array Approach

```python
numpy_array = np.array(range(1000000))

# Multiply entire array by 2
start_time = time.time()
result = numpy_array * 2
numpy_time = time.time() - start_time

# Result: ~0.005 seconds
```

NumPy is 30x faster in this example!
That's the difference between waiting 5 minutes vs 10 seconds for your analysis.

Creating NumPy Arrays

Multiple ways to create arrays for different use cases:

From Existing Data

```python
# From Python list
coffee_data = [3.2, 4.1, 2.8, 5.0]
coffee_array = np.array(coffee_data)

# From nested lists (2D)
papers_by_year = [[1, 2], [2, 3], [1, 4]]
papers_matrix = np.array(papers_by_year)
```

Built-in Constructors

```python
# Arrays of zeros/ones
zeros = np.zeros(10)
ones = np.ones(5)

# Range arrays
years = np.arange(1, 8)  # 1 to 7

# Random arrays
stress = np.random.normal(5, 2, 100)
```

Coursework Applications

Initialize parameter arrays, create homework data, set up optimization variables

Vectorized Operations: The Magic

Apply operations to entire arrays at once:

```
# PhD student data
papers_per_year = np.array([0, 1, 2, 1, 3, 2, 1])  # 7 years of PhD
conference_costs = np.array([1200, 1500, 800, 2000, 1100])

# Mathematical operations on entire arrays
total_papers = np.sum(papers_per_year)         # 10 papers total
avg_papers = np.mean(papers_per_year)          # 1.43 papers/year
productivity_doubled = papers_per_year * 2     # [0, 2, 4, 2, 6, 4, 2]

# Statistical operations
print(f"Conference costs: ${np.mean(conference_costs):.0f} ± ${np.std(
    conference_costs):.0f}")
```

No Loops Needed!
NumPy handles the iteration internally in optimized C code.

Boolean Indexing: Powerful Filtering

Filter data based on conditions:

```python
# Coffee consumption data
coffee_cups = np.array([2.1, 4.5, 8.2, 3.1, 12.5, 1.9, 6.8])

# Boolean conditions
high_caffeine = coffee_cups > 5.0
extreme_caffeine = coffee_cups > 10.0

# Filter the data
moderate_drinkers = coffee_cups[coffee_cups <= 5.0]
caffeine_addicts = coffee_cups[coffee_cups > 8.0]

# Count results
print(f"High caffeine consumers: {np.sum(high_caffeine)}")
```

Coursework Use Case

Filter homework results, identify outliers, select subsets for analysis

NumPy in Action: Statistical Analysis

Essential functions for data analysis:

Function	Purpose
`np.mean()`, `np.median()`	Central tendency
`np.std()`, `np.var()`	Variability
`np.min()`, `np.max()`	Range
`np.percentile()`	Quartiles and percentiles
`np.corrcoef()`	Correlation analysis
`np.unique()`	Count unique values
`np.where()`	Conditional selection

Real Research Example

Analyzing Hunt Library usage: mean=25.3 hours/week, std=12.1, 15% spend >40 hours/week

Pandas DataFrames

NumPy vs Pandas: Choose Your Tool

NumPy Arrays

✓ Homogeneous data
✓ Mathematical operations
✓ Maximum performance
✓ Numerical computations

Simulations, algorithms, math

Pandas DataFrames

✓ Mixed data types
✓ Labeled data (columns)
✓ Data cleaning tools
✓ Real-world datasets

Datasets, analysis, cleaning

Best of Both Worlds

Pandas is built on NumPy - you get speed + convenience!

Meet the DataFrame: Your Data Analysis Workhorse

DataFrame = Supercharged Spreadsheet

student_id	department	papers	coffee
PhD_001	ISE	3	4.2
PhD_002	OR	5	3.8
PhD_003	CSC	2	6.1
...	...	...	...

✓ Mixed data types ✓ Column labels ✓ Row indexing ✓ Missing value handling

Why DataFrames Matter

Real data is messy, mixed-type, and labeled - exactly what DataFrames handle best!

Essential DataFrame Exploration

First things first - understand your data:

Basic Inspection

```python
# Load and examine
df = pd.read_csv('phd_data.csv')

# Quick overview
df.shape              # (280, 11)
df.head()             # First 5 rows
df.tail()             # Last 5 rows
df.info()             # Data types, nulls
df.describe()         # Statistics
```

Understanding Structure

```python
# Column exploration
df.columns            # Column names
df.dtypes             # Data types
df.isnull().sum()     # Missing values

# Unique values
df['department'].unique()
df['department'].value_counts()
```

PhD Dataset

280 students, 11 variables, messy department names, missing coffee data (the horror!)

Data Selection: Getting What You Need

Multiple ways to slice and dice your data:

```
# Select columns
departments = df['department']                    # Single column (Series)
basics = df[['student_id', 'department']]        # Multiple columns (DataFrame)

# Filter rows
veterans = df[df['years_in_program'] >= 4]       # Boolean filtering
ise_students = df[df['department'] == 'ISE']     # Exact match

# Complex conditions
productive_ise = df[(df['department'] == 'ISE') &
                   (df['papers_published'] > 2)]  # Multiple conditions

# Statistical filtering
high_stress = df[df['stress_level'] > df['stress_level'].quantile(0.75)]
```

Coursework Applications

Filter by assignment conditions, select participant subgroups, analyze specific time periods

GroupBy: The Power of Aggregation

Analyze data by groups - essential for coursework:

```python
# Basic grouping
dept_analysis = df.groupby('department')['papers_published'].mean()

# Multiple aggregations
analysis = df.groupby('department').agg({
    'papers_published': ['mean', 'std'],
    'stress_level': 'mean',
    'coffee_cups_per_day': 'mean'
})

# Custom productivity score
def productivity_score(group):
    return (group['papers_published'] * 2 + group['conferences_attended']).mean()
```

Insight

OR students publish most papers (2.6 avg), but ISE students have better work-life balance!

String Operations: Cleaning Text Data

Our department names are a mess - let's fix them:

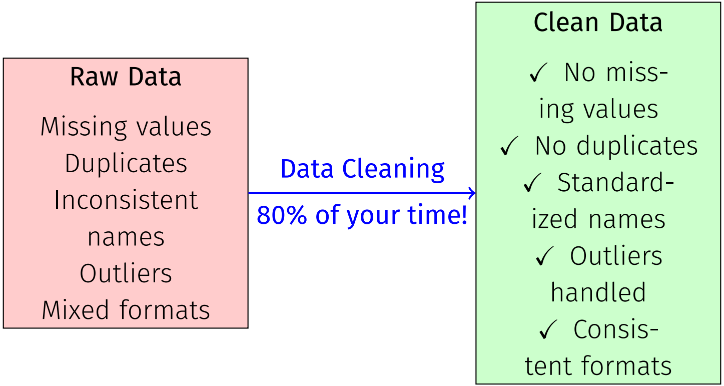
```python
# The problem
df['department'].unique()
# ['ISE', 'I.S.E.', 'Industrial Systems', 'OR', 'O.R.', ...]

# Cleaning function
def clean_department(dept):
    if 'Industrial' in dept or dept in ['ISE', 'I.S.E.']:
        return 'ISE'
    elif 'Operations' in dept or dept in ['OR', 'O.R.']:
        return 'OR'
    # ... more cleaning logic

# Apply and verify
df['department_clean'] = df['department'].apply(clean_department)
df['department_clean'].value_counts()
```

Data Cleaning Workshop

The Reality of Data



The 80/20 Rule
Data scientists spend 80% of their time cleaning data, 20% analyzing it.

Step 1: Handle Missing Values Strategically

Different strategies for different data types:

```python
# Assess the damage
missing_summary = df.isnull().sum()
# coffee_cups_per_day: 22 missing, stress_level: 14 missing, funding_amount: 8 missing

# Strategy 1: Fill with median (coffee - some students don't drink coffee)
coffee_median = df['coffee_cups_per_day'].median()  # 3.5 cups
df['coffee_cups_per_day'].fillna(coffee_median, inplace=True)

# Strategy 2: Fill with mean (stress - continuous variable)
stress_mean = df['stress_level'].mean()  # 5.8 out of 10
df['stress_level'].fillna(stress_mean, inplace=True)

# Strategy 3: Fill with group-specific values (funding by department)
funding_by_dept = df.groupby('department')['funding_amount'].mean()
df['funding_amount'] = df.apply(
    lambda row: funding_by_dept[row['department']] if pd.isna(row['funding_amount'])
    else row['funding_amount'], axis=1)
```

Step 2: Remove Duplicates

Duplicate data can skew your analysis:

```python
# Check for duplicates
print(f"Duplicate rows: {df.duplicated().sum()}")  # 8 duplicates found

# Examine duplicates (optional)
duplicates = df[df.duplicated(keep=False)].sort_values('student_id')
print(duplicates[['student_id', 'department', 'papers_published']].head())

# Remove duplicates (keep first occurrence)
original_shape = df.shape
df.drop_duplicates(inplace=True)
new_shape = df.shape

print(f"Removed {original_shape[0] - new_shape[0]} duplicate rows")
print(f"New dataset shape: {new_shape}")
```

Why This Matters

Duplicates can artificially inflate correlations and bias statistical tests.

Step 3: Handle Outliers Intelligently

Outliers: real extreme values or data entry errors?

```python
# Identify outliers using IQR method
Q1 = df['coffee_cups_per_day'].quantile(0.25)    # 2.1 cups
Q3 = df['coffee_cups_per_day'].quantile(0.75)    # 4.8 cups
IQR = Q3 - Q1                                     # 2.7 cups

# Define outlier bounds
lower_bound = Q1 - 1.5 * IQR  # Anything below -1.95 cups (impossible)
upper_bound = Q3 + 1.5 * IQR  # Anything above 8.85 cups (concerning!)

# Find extreme outliers
extreme_coffee = df['coffee_cups_per_day'] > 10  # 5 students drinking 10+ cups/day

# Cap extreme values (medical safety!)
df.loc[df['coffee_cups_per_day'] > 8, 'coffee_cups_per_day'] = 8
print("Capped extreme coffee consumption at 8 cups/day for student safety")
```

Domain Knowledge Matters

Statistical outliers aren’t always wrong - use research context to decide!

Step 4: Feature Engineering

Create new variables from existing data:

```
# Productivity score (papers worth 2x conferences)
df['productivity_score'] = (df['papers_published'] * 2 +
                           df['conferences_attended']) / df['years_in_program']

# Work-life balance indicator
df['work_life_balance'] = 10 - (df['hours_in_hunt_library_per_week'] / 10 +
                                df['stress_level']) / 2
df['work_life_balance'] = df['work_life_balance'].clip(1, 10)

# Categorical features
df['seniority'] = pd.cut(df['years_in_program'],
                        bins=[0, 2, 4, 10],
                        labels=['Early', 'Mid', 'Advanced'])

df['caffeine_level'] = pd.cut(df['coffee_cups_per_day'],
                             bins=[0, 2, 4, 8],
                             labels=['Low', 'Moderate', 'High'])
```

Feature Engineering Results

Our new features reveal insights:

Seniority Level	Count	Avg Productivity
Early (1-2 years)	84	1.2
Mid (3-4 years)	126	2.1
Advanced (5+ years)	70	2.8

Caffeine Level	Count	Avg Stress
Low (<2 cups)	45	4.2
Moderate (2-4 cups)	156	5.8
High (4+ cups)	79	7.1

Insight

More coffee = more stress! But also, advanced students are most productive.

Advanced Operations & Wrap-up

Correlation Analysis: What Drives Success?

Key correlations with papers published:

Variable	Correlation
Years in program	0.68
Advisor meetings per month	0.42
Hours in Hunt Library	0.31
Coffee consumption	0.18
Funding amount	0.15
Stress level	0.09
Distance from campus	-0.12

Research Insights

Experience matters most, but regular advisor meetings and library time also boost productivity!

Data Visualization Preview

Papers by Department

Bar chart showing OR leads with 2.6 papers/student

Coffee vs Papers

Scatter plot reveals weak positive correlation

Experience vs Output

Clear upward trend: time leads to productivity

Stress Distribution

Normal-ish distribution centered around 5.8/10

Coming Up in Block 3

We'll use this clean data to build predictive models with machine learning!

What We Accomplished in Block 2 ✓

- ✓ **Mastered NumPy** for fast numerical computation
- ✓ **Conquered Pandas** for data manipulation and analysis
- ✓ **Cleaned messy data** using professional techniques
- ✓ **Created new features** from existing variables
- ✓ **Prepared dataset** ready for machine learning

Dataset Transformation
From 288 messy rows with missing values → 280 clean rows ready for analysis!

Preview: Block 3 (11:00 AM - 11:50 AM)

From Data to Insights: Predictive Modeling

What's Coming

- **Machine learning** fundamentals
- **Linear regression** with scikit-learn
- **Model evaluation** and interpretation
- **Predicting** PhD student success
- **Feature importance** analysis

Your Clean Data in Action

- Train models on our cleaned dataset
- Predict papers published
- Understand what drives success
- Build your first ML pipeline

10-minute break, then we build models!

Key Takeaways

Technical Skills

- **NumPy** for computational speed
- **Pandas** for data manipulation
- **Data cleaning** strategies
- **Feature engineering** techniques
- **Exploratory analysis** methods

Research Mindset

- Always explore data first
- Clean systematically
- Document your decisions
- Create meaningful features
- Validate your cleaning

Remember: Your dataset is saved as 'phd_research_productivity_clean.csv'

Ready for machine learning in Block 3!

Questions?

See you in 10 minutes for Block 3!